

Sybase, Inc.
One Sybase Drive
Dublin, CA 94568
www.sybase.com

Sybase Avaki EII Provisioning and Advanced Data Integration Guide

Release 7.0 • August 24, 2006

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase, SYBASE (logo), ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Advantage Database Server, Afaia, Answers Anywhere, Applied Meta, Applied Metacomputing, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-Translator, APT-Library, ASEP, Avaki, Avaki (Arrow Design), Avaki Data Grid, AvantGo, Backup Server, BayCam, Beyond Connected, Bit-Wise, BizTracker, Certified PowerBuilder Developer, Certified SYBASE Professional, Certified SYBASE Professional Logo, ClearConnect, Client-Library, Client Services, CodeBank, Column Design, ComponentPack, Connection Manager, Convoy/DM, Copernicus, CSP, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DataWindow .NET, DB-Library, dbQueue, Dejima, Dejima Direct, Developers Workbench, DirectConnect Anywhere, DirectConnect, Distribution Director, Dynamic Mobility Model, e-ADK, E-Anywhere, e-Biz Integrator, E-Whatever, EC Gateway, ECMAP, ECRTP, eFulfillment Accelerator, EII Plus, Electronic Case Management, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise Portal (logo), Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, eremote, Everything Works Better When Everything Works Together, EWA, ExtendedAssist, ExtendedView, Financial Fusion, Financial Fusion (and design), Financial Fusion Server, Formula One, Fusion Powered e-Finance, Fusion Powered Financial Destinations, Fusion Powered STP, Gateway Manager, GeoPoint, GlobalFIX, iAnywhere, iAnywhere Solutions, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InstaHelp, Intelligent Self-Care, InternetBuilder, iremote, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, Legion, Logical Memory Manager, IRLite, M2M Anywhere, Mach Desktop, Mail Anywhere Studio, Mainframe Connect, Maintenance Express, Manage Anywhere Studio, MAP, M-Business Anywhere, M-Business Channel, M-Business Network, M-Business Suite, MDI Access Server, MDI Database Gateway, media.splash, Message Anywhere Server, MetaWorks, MethodSet, mFolio, Mirror Activator, ML Query, MobiCATS, MobileQ, MySupport, Net-Gateway, Net-Library, New Era of Networks, Next Generation Learning, Next Generation Learning Studio, O DEVICE, OASIS, OASIS logo, ObjectConnect, ObjectCycle, OmniConnect, OmniQ, OmniSQL Access Module, OmniSQL Toolkit, OneBridge, Open Biz, Open Business Interchange, Open Client, Open ClientConnect, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, Partnerships that Work, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, Pharma Anywhere, PhysicalArchitect, Pocket PowerBuilder, PocketBuilder, Power++, Power Through Knowledge, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, Powering the New Economy, PowerScript, PowerSite, PowerSocket, Powersoft, PowerStage, PowerStudio, PowerTips, Powersoft Portfolio, Powersoft Professional, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Pylon, Pylon Anywhere, Pylon Application Server, Pylon Conduit, Pylon PIM Server, Pylon Pro, QAnywhere, Rapport, Relational Beans, RemoteWare, RepConnector, Report Workbench, Report-Execute, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Resource Manager, RFID Anywhere, RW-DisplayLib, RW-Library, SAFE, SAFE/PRO, Sales Anywhere, Search Anywhere, SDF, Search Anywhere, Secure SQL Server, Secure SQL Toolset, Security Guardian, ShareLink, SharePool, SKILS, smart.partners, smart.parts, smart.script, SOA Anywhere Trademark, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL SMART, SQL Toolset, SQL Server/CFT, SQL Server/DBM, SQL Server SNMP SubAgent, SQL Station, SQLJ, Stage III Engineering, Startup.Com, STEP, SupportNow, S.W.I.F.T. Message Format Libraries, Sybase Central, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Financial Server, Sybase Gateways, Sybase Learning Connection, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase Virtual Server Architecture, Sybase User Workbench, SybaseWare, Syber Financial, SyberAssist, SybFlex, SybMD, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, The Enterprise Client/Server Company, The Extensible Software Platform, The Future Is Wide Open, The Learning Connection, The Model For Client/Server Solutions, The Online Information Center, The Power of One, TotalFix, TradeForce, Transact-SQL, Translation Toolkit, Turning Imagination Into Reality, UltraLite, UltraLite.NET, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, Viafone, Viewer, VisualWriter, VQL, WarehouseArchitect, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server, XcelleNet, XP Server, XTNDAccess and XTNDConnect are trademarks of Sybase, Inc. or its subsidiaries. 07/06

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names used herein may be trademarks or registered trademarks of their respective companies.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Set in *Arial*, *Courier New*, and *Times New Roman*. Stanley Morison, the creator of *Times New Roman*, said of it: "By the vice of Mammon and the misery of the machine, it is bigoted and narrow, mean and puritan."

Credits

This product includes software developed by the Apache Software Foundation (<http://www.apache.org>). This product includes Hypersonic SQL and ANTLR. This product includes code licenses from RSA Security, Inc. Some portions licensed from IBM are available at <http://oss.software.ibm.com/icu4j/>. Contains IBM® 64-bit Runtime Environment for AIX™, Java™ 2 Technology Edition Version 1.4 Modules © Copyright IBM Corporation 1999, 2000 All Rights Reserved. Contains the SAXON XSLT Processor from Michael Kay, which is available at <http://saxon.sourceforge.net>. This product includes software developed by the Proxool Project (<http://proxool.sourceforge.net>).

Sybase Avaki EII Provisioning and Advanced Data Integration Guide
Written by Cheryl Magadieu, Beth Thoenen, and Ed Blachman

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

Table of contents

Preface **vii**

Organization **viii**

Related documentation and online help **ix**

Conventions **xi**

 Command syntax conventions **xi**

 Conventions for screen examples **xii**

How to contact Avaki support at Sybase, Inc. **xiii**

Chapter 1

Managing information from databases **1**

Database overview **1**

 About database operations **1**

 Uses of database operations **2**

Connecting to databases **3**

 Creating database connectors **3**

 Viewing and modifying database connectors **8**

 Viewing database schemas **10**

 Viewing associated database operations **14**

 Setting database operation permissions **14**

 Testing database connectors **20**

 Managing SQL views **20**

 Removing database connectors **21**

Managing database operations **22**

 Creating database operations **22**

 Viewing and modifying database operations **29**

 Viewing database operation details **29**

 Managing database operation metadata **31**

 Executing database operations **36**

 Removing database operations **38**

Managing SQL views **39**

 Provisioning SQL views **39**

 Viewing SQL views **43**

 Modifying SQL views **43**

Removing SQL views	44
Configuring SQL view attributes	45
Managing SQL view categories	48

Chapter 2

Basic data integration 49

Virtual database operations	50
Creating virtual database operations	50
Viewing and modifying virtual database operations	56
Managing metadata for virtual database operations	57
Executing virtual database operations	62
Removing virtual database operations	63
Managing virtual database services	64
Browsing virtual database schemas	64
Configuring virtual database access permissions	65
Configuring virtual database service attributes	71
Managing virtual database service ACLs	73
Data services overview	75
About data services	75
Understanding data service components	76
Avaki Studio and data services	78
Data services and distributed transactions	78
Creating data services	81
Getting started	81
Configuring data service plug-ins	82
Configuring data service input parameters	84
Configuring data service output streams	86
Configuring data service input streams	87
Importing data service descriptors	92
Viewing a list of data services	94
Modifying data services	94
Managing data service metadata	97
Viewing data service dependencies	97
Generating a data service's schema	99
Exposing data service results as a SQL view	101
Testing data services	102
Removing data services	103

Chapter 3

Managing cache services 105

- Configuring clients and Avaki servers to use cache services **106**
- Configuring caching for files **107**
 - On-demand caching **107**
 - Pinning files for scheduled caching **108**
 - Permissions and access control **108**
- Configuring caching for database operations and data services **108**
 - On-demand caching **108**
 - Scheduled caching **109**
 - Remote/local caching interactions **110**
 - Permissions and access control **110**
- Associating Avaki servers with caches **111**
 - Associating grid servers with caches **111**
 - Disassociating grid servers from caches **112**
 - Associating data grid access servers with caches **113**
 - Disassociating data grid access servers from caches **115**
 - Viewing and modifying cache service configuration **116**
- Overriding cache service default settings **119**
- Managing caches **120**
 - Managing file or directory caches **120**
 - Managing database caches **139**
 - Managing data service caches **152**
 - Configuring schedule exclusions **166**

Chapter 4

Setting up data service plug-ins 175

- Overview of data service plug-ins **176**
- Java, JavaScript, or XSLT **176**
- Input and output **177**
 - Input sources **177**
 - Parameters **178**
 - Output stream **180**
- Plug-in files **180**
 - JAR files and manifest files for Java plug-ins **180**
 - JavaScript file for JavaScript plug-ins **180**
 - XSL file for XSLT plug-ins **180**
- Deployment of plug-ins **180**
- Creating XSLT plug-ins **181**
 - Specifying parameters **181**

Specifying secondary input sources	181
A sample XSLT plug-in	182
Creating Java plug-ins with the Plug-in Wizard	183
Prerequisites	184
Plug-in Wizard procedure	184
Writing the Java code	185
Creating JavaScript plug-ins	200
Access to Java classes and interfaces	200
Import required packages	201
Methods available on the plug-in object	202
Execute function	203

Chapter 5

Provisioning web services 205

Provisioning web services overview	205
Setting up the data service	206
Getting started	207
Specifying parameters	207
Specifying the output stream	208
Specifying the input streams	208
Specifying a grid server	213
Testing data services	214

Chapter 6

Managing views 217

Managing view generators	218
Setting up file view generators	218
Setting up database operation view generators	221
Setting up data service view generators	225
Displaying view dependencies	228
Modifying view generators	229
Generating views	240
About generated view files	240
Cache interactions	241
View generation procedure	241
Non-XSLT-based view generators	242
The TrAX standard	243
Implementing a Java transformer	243
Installing your Java transformer	245
Using your transformer	245

	Referring to other documents in your transformer	245
	Logging errors	246
Appendix A	<i>Advanced database management</i>	247
	Configuring the JDBC driver JAR file path	247
	Restricting database operation output	249
	Configuring batch mode settings	250
	Configuring SQL calls	251
	Configuring database operation timeouts	253
	Configuring database operation fetch size	254
Appendix B	<i>Data service schema</i>	257
Appendix C	<i>Data representation in Avaki</i>	273
	Rowset objects	274
	Rowsets and XML as inputs	275
	Usage scenarios	275
Appendix D	<i>Avaki rowset XML</i>	277
	Core schema	277
	Rowset-specific schema	280
	Sample XML schema for a database operation	280
	<i>Glossary</i>	289

Preface

This *Sybase Avaki EII Provisioning and Advanced Data Integration Guide* describes how to use the Sybase Avaki EII web user interface to provision and integrate data from files and databases. The guide is intended for anyone who exposes, manipulates, or caches data.

Read the *Avaki Overture* for an introduction to concepts and tools discussed in this book, including database connectors, database operations, SQL views, data services, and cache services. To learn how to use Avaki's other interfaces (Avaki Studio and the Avaki command-line interface) for provisioning and integration, read *Data Integration with Sybase Avaki Studio* or the *Sybase Avaki EII Command Reference*.

Note This book and the product's user interfaces refer to Sybase Avaki EII software as *Avaki* or *Avaki Data Grid*.

Organization

This book is organized as follows:

Chapter 1 Managing information from databases	Describes how to create and manage database connections, database operations, and SQL views.
Chapter 2 Basic data integration	Explains how to integrate data from multiple, heterogeneous, distributed data sources, such as files, relational data, XML data, and application data.
Chapter 3 Managing cache services	Explains how to add files to a cache, unpin files, add database operations to a cache, schedule and unschedule database actions, and view caching services.
Chapter 4 Setting up data service plug-ins	Explains how to write data service plug-ins in Java, JavaScript, or XSLT.
Chapter 5 Provisioning web services	Explains how to provision data from a web service into a grid.
Chapter 6 Managing views	Describes how to set up and use views that use database operations and files as input sources.
Appendix A Advanced database management	Describes some advanced settings that you can configure for database connectors and database operations
Appendix B Data service schema	Describes the XML schema that specifies the contents of a data service.
Appendix C Data representation in Avaki	Describes how Avaki represents data using rowsets and XML.
Appendix D Avaki rowset XML	Describes the schema that Avaki rowset XML comprises.
Glossary	Defines terms used in this guide.

Related documentation and online help

Manuals

These manuals make up the Avaki documentation set:

- *Sybase Avaki EII Overture*
- *Sybase Avaki EII Administration Guide* (includes installation instructions)
- *Data Integration with Sybase Avaki Studio*
- *Sybase Avaki EII Provisioning and Advanced Data Integration Guide*
- *Sybase Avaki EII API Guide*
- *Sybase Avaki EII Command Reference*

The manuals are included, in PDF format, on the CD with the Avaki software. They are stored in the docs subdirectory of the Avaki installation directory.

To access the manuals via Avaki's web user interface, log in to your Avaki domain and click the **Help** link at the top right corner of any page of the web UI.

Online help

In addition to the manuals, Avaki provides online help for commands.

To display a list of Avaki commands with brief descriptions, log in to the Avaki system and enter **avaki help**:

```
% avaki help
List of domain commands:
attribute
backup
cache
cat
categories
cd
chmod
chown
client
cp
dataservice
dbconn
dbop
dgas
```

```
directory
domain
executionservice
file
group
help
id
ldap
ln
locks
login
logout
ls
mkdir
monitor
mv
nis
passwd
patch
permissions
plugin
proxy
pwd
replica
rm
scheduleexclusion
search
security
server
share
shell
sqlview
status
systemproperty
upgrade
user
view
virtualdatabase
virtualschema
whoami
```

To display a description of a particular command and the syntax, enter a command of the form **avaki help** *<command>*. For example:

```
% avaki help mv
usage: avaki mv <source-grid-path> <target-grid-path>
Description: Move or rename a grid directory or a file
in a grid directory. Similar to the Unix mv command.
```

Conventions

This section describes text conventions used in this guide to represent elements of commands and screen displays.

Command syntax conventions

This table describes conventions that this book uses in command syntax statements. The “Enter this” column tells you whether you need to enter the characters when you type a command. The examples in the “Examples” column are not necessarily complete commands.

Convention	Description	Enter this?	Examples
[]	Square brackets surround optional arguments.	no	<code>avaki login [<user-id>]</code>
{ }	Curly brackets surround groups of required arguments.	no	<code>avaki chmod {--allow --deny --unset}</code>
{ } []	Vertical bars separate alternative options within square or curly brackets. If the brackets are square, you need not enter any of the options; if the brackets are curly, you must choose one of the options.	no	<code>avaki backup {--snapshot --recover}</code> <code>avaki share --create [--background --bg]</code>
< >	Angle brackets surround placeholder arguments that you must replace with a value such as a path or file name. Square brackets outside the angle brackets indicate that the placeholder is optional.	no	<code>avaki help [<command-name>]</code> <code>avaki help share</code>
*	An asterisk follows an argument that can be entered zero or more times.	no	<code>avaki plugin --generate [--input=<stream-spec>*]</code> <code>avaki plugin --generate --input="name=input1,type=XML"</code>
+	A plus sign follows an argument that can be entered one or more times. Use spaces to separate the values.	no	<code>avaki cat <grid-path>+</code> <code>avaki cat /home/fred/file1 /home/fred/file2</code>

Convention	Description	Enter this?	Examples
-	Enter a hyphen or minus sign before a single-letter command option.	yes	avaki mkdir -p
=	Enter an equal sign before the value of an option.	yes	avaki login --auth-service= <auth-service-name>
(space)	A space separates multiple arguments.	yes	avaki cat file1 file2 file3

Conventions for screen examples

This table describes conventions this book uses in examples of user input and system output.

Convention	Description	Example
\$ or C:>	The command prompt	\$
< >	A placeholder; replace the content inside the brackets with an option or value	\$ avaki ls <grid-path>
screen font	Text that appears on the screen	sample text
bold screen font	User input—commands that you enter	\$ avaki ls

How to contact Avaki support at Sybase, Inc.

For general information about Sybase technical support, see the *Customer Service Reference Guide* at

<http://www.sybase.com/support/aboutsupport/guide/csrg>

Please contact us with any questions or difficulties you encounter.

By telephone

In North America, call toll free: 1-800-8SYBASE

Outside North America, follow the link below to see a list of Sybase offices and phone numbers around the world.

<http://www.sybase.com/contactus/support>

On the web

If you are a designated contact for a technical support plan, you can log and track cases on the web using the Case Express application. At www.sybase.com, mouse over the **Support and Services** tab and select **Case Management** from the drop-down list. Use the email address and password for your mysybase account to log in.

Managing information from databases

This chapter describes how to create, view, test, and remove database connectors; how to create, view, execute, schedule, and remove database operations; and how to provision, view, and modify SQL views. All these tasks fall under the heading of provisioning—getting information into the data grid.

The chapter covers the following topics:

- [“Database overview,”](#) below
- [“Connecting to databases”](#) on page 3
- [“Managing database operations”](#) on page 22
- [“Managing SQL views”](#) on page 39

Database overview

About database operations

In Avaki, a *database operation* is the vehicle through which users, applications, and Avaki objects such as data services and generated SQL views have access to data in relational databases. Each database operation accesses one relational database. The data owner (typically a database administrator who has responsibility for that database) creates the database operation as an entry in the Avaki data catalog, giving it a name and a definition that is a SQL statement, such as calling a stored procedure. This

can be any SQL statement the database will accept, including row operations (INSERT, SELECT, UPDATE, DELETE, CALL) or any other SQL, but it is most often a query represented by a SELECT statement. When you define a database operation that contains a SQL statement and grant access to that database operation, you are granting rights to run that statement in the database.

Database operations can accept and return parameters. For example, you can set up a statement that returns order details given a specific order number, and then supply the order number at runtime.

The mechanism that enables a database operation to connect with a database is called a *database connector*. You create the database connector before creating the database operation. For more on database connectors, see [“Connecting to databases” on page 3](#).

Uses of database operations

Once you have one or more database operations, you can use them as input sources or cache the results for later use:

Database operations as input sources. Database operations (including virtual database operations) can be invoked directly via ODBC, JDBC, the Avaki web service interface or the various Avaki user interfaces. But they can also be used within Avaki by other Avaki operations:

- **Data services.** A data service combines and/or transforms data obtained from various sources, including database operations. For more on data services, see [Chapter 2, “Basic data integration” on page 49](#).

A special application of data services is to use the two-phase commit protocol to execute several database operations as a single *distributed transaction*. For more on distributed transactions, see [“Data services and distributed transactions” on page 78](#).

Avaki Studio provides powerful, easy-to-use tools for building and deploying Avaki data services. For more on Avaki Studio, see *Data Integration with Sybase Avaki Studio*.

- **View generators.** A view generator obtains data from a grid file, a data service, or a database operation, and saves the results as a file. For more information, see [Chapter 6, “Managing views” on page 217](#).
- **SQL views.** You can generate a SQL view—a “virtual table”—from a database operation. This allows the results of that database operation to be accessed via ODBC or JDBC by applications (or users) that are more comfortable dealing with

Avaki as a collection of tables than as a collection of callable procedures. For more information, see [“Managing SQL views” on page 39](#).

Caching database operations. You can store the results of a database operation or a virtual database operation in a cache to reduce the load on a back-end data source or cut down on network congestion and speed up application performance. For details about caching database operation results, see [Chapter 3, “Managing cache services”](#).

Connecting to databases

A *database connector* enables one or more database operations or ad-hoc queries to connect to a relational database. Database connectors contain information about how to access a particular database. Once you’ve created a database connector, you can create database operations that use the database connector to access a database. Database operations can insert new data into or update or delete existing data in a relational database. They can also call stored procedures in such a database, or extract data from the database and deliver it to an Avaki view generator, data service, or application client (ODBC/JDBC/SOAP).

This section covers the following topics:

- [“Creating database connectors,”](#) below
- [“Viewing and modifying database connectors” on page 8](#)
- [“Viewing associated database operations” on page 14](#)
- [“Setting database operation permissions” on page 14](#)
- [“Testing database connectors” on page 20](#)
- [“Removing database connectors” on page 21](#)

Creating database connectors

Follow these steps to create a database connector:

- Step 1** Log in as a member of the DatabaseAdministrators group.
- Step 2** Navigate to the Create Database Connector screen:

Home > Database provisioning > Create database connector

Create Database Connector

To create a new database connector, fill in the information below, and then click **Submit**.

Connector name:

Allow ad-hoc queries, schema browsing, and SQL view provisioning:

Description (optional):

Database name (optional):

Database driver:

Default connection username (optional):

Default connection password (optional):

Confirm default connection password (optional):

JDBC fetch size: Default
 Custom:

Allow database identity mappings:

Database administrator's full name (optional):

Connection string:

Grid server: ▼

Use Avaki connection pooling: Yes, with the default pool size (15)
 Yes, with a custom pool size:
 No

Connection properties (optional):

XA driver (optional):

XA connection properties (optional):

Step 3 Fill in the form:

- Connector name: Enter a name for the database connector. **Note:** Do not include any spaces in the name.

- Allow ad-hoc queries, schema browsing, and SQL view provisioning: Select this option if you want to enable users to perform direct SQL queries against the database, browse the database's metadata, or provision SQL views. Ad-hoc queries must run through an existing Avaki database connector. Ad-hoc queries can be thought of as single-use database operations. You can run an ad-hoc query using either the CLI or the JDBC driver. For information about using the CLI to run an ad-hoc query, see the *Sybase Avaki EII Command Reference*. For information about using a JDBC driver to run an ad-hoc query, see the *Sybase Avaki EII API Guide*.
- Description (optional): Enter some descriptive information about this database connector.
- Database name (optional): Enter the name of the database.
- Database driver: Enter the class name of your database JDBC driver. For example:

```
com.sybase.jdbc3.jdbc.SybDriver
```

```
oracle.jdbc.driver.OracleDriver
```

Make sure that the driver that you specify is in the following directory:

```
<Avaki-install-dir>/drivers
```

For more information about configuring JDBC drivers, see the *Sybase Avaki EII Administration Guide*.

- Default connection username (optional): Enter the name of a user account in the database. This username will be used to authenticate the database connection. By default, all database operations associated with this database connector will use this username when executed.
- Default connection password (optional): Enter the password for the default connection username.
- Confirm default connection password (optional): Confirm the connection password. **Note:** while the password is optional, if you choose to supply one, you must confirm it here.
- JDBC fetch size: This parameter can be used to fine-tune performance of database operations. When an application uses the JDBC driver to execute a database operation, it typically processes the rows that are returned one after another; but the driver applies a buffering optimization by fetching rows in batches; the fetch size is the number of rows to be fetched in such a batch. In most circumstances, the driver's default fetch size will be optimal, so you'll want to keep the Default setting. But if you decide that database operations executed through this database con-

connector should generally use a non-default fetch size, enter the relevant number of rows in the Custom text box. **Note:** you can also set the fetch size for individual database operations. See [“Configuring database operation fetch size” on page 254](#).

- Allow database identity mappings: Place a check mark in this box if database identity mappings are allowed on this database connector. A database identity mapping is a special-purpose user alias. It allows operations performed by some Avaki user on a particular database connector to be executed in the name of an alternate user/password combination. For details about configuring database identity mappings, see the *Sybase Avaki EII Administration Guide*.
- Database administrator’s full name (optional): Enter the database administrator’s first and last name.
- Database administrator’s email address (optional): Enter the database administrator’s email address.
- Database administrator’s phone number (optional): Enter the database administrator’s telephone number.
- Database administrator’s organization (optional): Enter the organization to which the database administrator belongs.
- Connection string: Enter the URL for your JDBC driver. For more information, see the documentation for your database. Here are two sample connection strings:

```
jdbc:sybase:Tds:gallium:15000/test
```

```
jdbc:oracle:thin:@gallium:1521:test1
```

- Grid server: Select the grid server on which to create the database connector.
- Use Avaki connection pooling: Specify whether to enable Avaki connection pooling. When connection pooling is enabled, database connections can be reused, which typically improves performance for JDBC applications. By default, Avaki connection pooling will keep up to 15 connections open to a back-end database. If your driver has built-in connection pooling, it may not be necessary to use Avaki connection pooling. Select one of the following options:
 - Yes, with the default pool size (15): Enable connection pooling, with a connection pool size of 15 connections.
 - Yes, with a custom pool size: Enable connection pooling and specify the desired connection pool size.
 - No: Do not enable connection pooling.

- Connection properties (optional): Specify any properties that are required for your database. For information about database-specific properties, see your database documentation.
- XA driver (optional): Specify an XA driver class if you plan to use this database connector to support distributed transactions. This class must reside in the JAR file for the database vendor's JDBC driver. These XA drivers have been tested with Avaki:
 - For Sybase ASE: `com.sybase.jdbc3.jdbc.SybXADataSource`
(ASE 15.0 with jConnect 6.05)
 - For Oracle 10g: `oracle.jdbc.xa.client.OracleXADataSource`
(Oracle 10g release 10.1.0.2.0 with JDBC driver version 10.2.0.1.0)
 - For MySQL: `com.mysql.jdbc.jdbc2.optional.MysqlXADataSource`
(MySQL 5.0 with MySQL Connector/J 5.0)
- XA connection properties (optional): Some XA connection properties might be required by your XA driver. Here are some that are typically specified:
 - For Sybase ASE:
 - `ServerName=<database-host-name>`
 - `PortNumber=<database-port>` (e.g. 5000)
 - `ResourceManagerType=2`
 - `ResourceManagerName=connection`
 - `DatabaseName=<database-name>`
 - `NetworkProtocol=Tds`
 - `User=<db-user-name>`
 - `Password=<db-user's-password>`
 - For Oracle 10g:
 - `URL=jdbc:oracle:thin:@<db-host-name>:<db-port>:<db-name>`
 - `User=<db-user-name>`
 - `Password=<db-user's-password>`
 - For MySQL:
 - `URL=jdbc:mysql://<db-host-name>[:<db-port>]/<db-name>`
 - `User=<db-user-name>`
 - `Password=<db-user's-password>`

Note For details on configuring your XA-compliant JDBC driver, including the particular XA connection properties to use, refer to the documentation for the driver.

- Step 4** Click **Submit**. The system displays a confirmation screen.
- Step 5** Optional. To ensure that the database connector works, click the **Test** button. Avaki tests both the regular JDBC connection to the database and the XA connection, if one is configured. The system displays a message indicating whether the connector is operational.

Test Database Connector

The connection **MyDBConnector** is operational.

The following information was obtained from the database:

```
Database system type: Oracle
Database version: Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options
JDBC Driver type: Oracle JDBC driver
JDBC Driver version: 9.2.0.1.0
```

For information about configuring advanced database connector settings, see [Appendix A, “Advanced database management”](#).

Viewing and modifying database connectors

To view a list of the database connectors in the current grid domain, navigate to the View Database Connectors screen:

Home > Database provisioning > Manage database connectors

View Database Connectors

Grid domain:

You are viewing the database connectors in the current domain.

Name	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security
MyDBConnector	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security

The system displays a list of the database connectors in the current Avaki domain.

To view details about a database connector, do the following:

Step 1 Navigate to the View Connector Detail screen:

Home > Database provisioning > Manage database connectors

Step 2 Click the **View/Edit** link beside the name of the database connector whose details you want to view. The Database Connector Detail screen appears.

Database Connector Detail

Connector name: CherylDomain.MyDBCConnector

Allow ad-hoc queries, schema browsing, and SQL view provisioning:

Source schema:

Description:

Database name:

Database driver:

Default connection username:

New default connection password:

Confirm new default connection password:

JDBC fetch size: Default
 Custom:

Allow database identity mappings:

Database administrator's full name:

Database administrator's email address:

Database administrator's phone number:

Database administrator's organization:

Connection string:

Grid server: blachmanxp.sybase.com

Use Avaki connection pooling: Yes, with the default pool size (15)
 Yes, with a custom pool size:
 No

Connection properties:

Step 3 Optional. Modify the database connector settings as needed, then click **Submit** to save your changes.

Viewing database schemas

Sometimes it is useful to be able to browse the schema—the set of table definitions—of the database that underlies a database connector. The pop-up Database Table Browser allows you to do this.

Database schemas are not all alike. The JDBC standard is written in terms of a three-level database table hierarchy: catalogs, which contain schemas, which in turn contain tables. But the standard allows each database to have its own name for the catalog and schema levels. And beyond that, databases differ as well regarding whether they implement all three levels or just two (and if the latter, whether it's the catalog or the schema level that is implemented), and in subtler ways besides. The look and feel of the Database Table Browser therefore varies subtly as well.

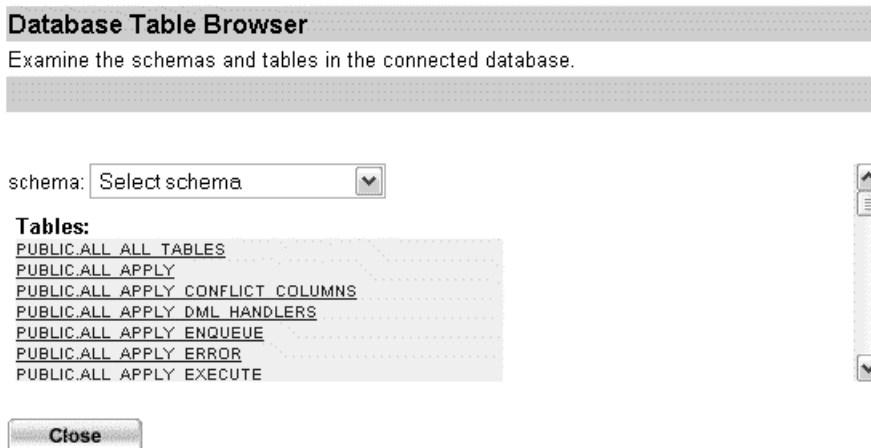
The Database Table Browser can be found in a number of ways. Here's one:

Step 1 Navigate to the View Connector Detail screen:

Home > Database provisioning > Manage database connectors

Step 2 Click the **View/Edit** link beside the name of the database connector whose details you want to view. The Database Connector Detail screen appears.

Step 3 Click the **Browse** button beside the Source schema field on the Database Connector Detail page. When you click the **Browse** button, the Database Table Browser window appears.



- In the database illustrated above, only two levels are implemented—schemas and tables. This database uses the word “schema” to correspond to the JDBC schema concept. The pull-down menu is used to select a particular schema or all schemas. A database that implements the catalog level rather than the schema level would look very similar, except that there's no “all catalogs” choice in a catalogs pull-

down. A database that implements all three levels will have two pull-downs rather than one.

- Even with no schema selected, the database illustrated above returns the list of all its tables. Not all databases do so; in one that did not, you'd see a message like

```
No tables can be seen with the current choice of schema.
```

rather than the list of table name links.

Step 4 Optional but recommended. Use the pull-down menu(s) to select a schema (or catalog, or combination of catalog and schema) to examine. Again, the result may either be a list of tables on the left or the “No tables can be seen” message, depending on

- whether the database is willing to show a list of tables (for the case where both catalog and schema levels are implemented but you've chosen only one), or

- whether the specified schema (or catalog, or schema-catalog combination) actually contains tables (it's quite possible that it does not).

Database Table Browser

Examine the schemas and tables in the connected database.

schema: ▼

Tables:

ANSWER
BIN\$1hJvHzN0RzaRSIpm2wPd6w==f0
BIN\$32VWn0F6QZw5Ww0c1ZPA9Uw==f0
BIN\$A7cklpsQ5y1rYyXY6H7zQ==f0
BIN\$CnxeiJQZTF0LWb6QfhIGlw==f0
BIN\$LhQ+vv7GTOCnii/G3q3xCg==f0
BIN\$Q36+Pdk9TsesNtrSBACYA==f0
BIN\$Y3+WaEi8TKCczTrNKv3uMA==f0
BIN\$YVQFZgnARiazv5I+CjzZow==f0
BIN\$aT5CdJzpTPCS9D0tc55fzw==f0
BIN\$foKxqX19QpGBBILy6JJq3Q==f0
BIN\$h6yMHfutQH+KTIsvAeWzZA==f0
BIN\$hc/aYrw5QXv0dgvae8elq==f0
BIN\$hD8ttYpxSC+SxgnoAsplCw==f0
BIN\$04+Q/RRmThyl8hevhFMUUQ==f0
BIN\$sAPhUH9ShuUINDpranWwQ==f0
BIN\$vczFtiMFQJG5vRK7RtoH5g==f0
BIN\$wwwrrBr6T5isdajgK9mgw==f0

BIRTHDAYS
BONUS
CATEGORIESTESTDB
DEPT
DOUBLETEST
EMP
EMP2
FLOATTEST
GEOGRAPHY
HELP

Close

Step 5 Click on the name of a table to see details of its definition.

Database Table Browser

Examine the schemas and tables in the connected database.

schema:

Tables:

ANSWER
 BIN\$1hJVhzN0RzaRSIpm2wPd6w==\$0
 BIN\$32VWn0F6QZw5Ww0c1ZPA9Uw==\$0
 BIN\$A7ck/lpsQ5v1YyXY6H7zQ==\$0
 BIN\$CnxeiJQZTFOLWb6QthlGlw==\$0
 BIN\$LhQ+yv7GT0Cnij/G3g3xCg==\$0
 BIN\$Q36+PdK9T SesNtsrSBACYA==\$0
 BIN\$Y3+WaEi8TKCczTrNKv3uMA==\$0
 BIN\$YvQFZgnARiazv5t+CizZow==\$0
 BIN\$aT5CdJzpTPCS9DDto55fzw==\$0
 BIN\$f0KxjX19QpGBBILy6JJq3Q==\$0
 BIN\$h6vMHfutQH+KTISvAeWzZA==\$0
 BIN\$hC/aYmw5QXyQdgyeah8elg==\$0
 BIN\$hD8tYpxSC+SxgncAspICw==\$0
 BIN\$o4+Q/RRmThvl8hevhFMUUQ==\$0
 BIN\$APhUH/9ShuUINDpranVWwQ==\$0
 BIN\$vozfFtjMFQJG5vRk7RtoH5g==\$0
 BIN\$wwrrB6T5iSdair3K9mgw==\$0
 BIRTHDAYS
 BONUS
 CATEGORIESTESTDB
 DEPT
 DOUBLETTEST
 EMP
 EMP2
 FLOATTEST
 GEOGRAPHY
 HELP

Table name: **SCOTT.DEPT**

Column Metadata

Name	Type	Size	Scale	Nullable
DEPTNO	DECIMAL	2	0	false
DNAME	VARCHAR	14	None	true
LOC	VARCHAR	13	None	true

Primary Key Columns

Column	Sequence
DEPTNO	1

Tables with Foreign Key References to DEPT

EMP

Close

Step 6 When you're done viewing the Database Table Browser, click **Close** to close the window.

Viewing associated database operations

To view the database operations that are associated with a particular connection, do the following:

- Step 1** Navigate to the View Database Connectors screen:

Home > Database provisioning > Manage database connectors

View Database Connectors

Grid domain:

You are viewing the database connectors in the current domain.

Name	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security
MyDBConnector	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security

- Step 2** Click the **Operations** link for the database connector whose database operations you want to see. The View Database Operations screen appears, showing the database operations that are associated with the database connector.

View Database Operations

You are viewing the database operations registered for the connection **MyDBConnector**.

Name	View/Edit	Execute	Schedule	Remove	Create View	Attributes	Security	Metadata
MyDBOperation	View/Edit	Execute	Schedule	Remove	Create View	Attributes	Security	Metadata

For details about managing database operations, see [“Managing database operations” on page 22](#).

Setting database operation permissions

Database operations can be created by a database administrator or by another user or group if the database administrator gives a user or group the appropriate permission. When a user or group creates a database operation, that database operation is owned by the database administrator who created the related database connector, so that the administrator can delete the database operation if necessary.

The following sections describe how to set database operation permissions:

- “Allowing users to create database operations,” below
- “Allowing groups to create database operations” on page 16
- “Preventing users from creating database operations” on page 18
- “Preventing groups from creating database operations” on page 19

Allowing users to create database operations

To allow a user to create a database operation that uses a designated database connection, the owner of the database connection must do the following:

Step 1 Navigate to the View Database Connectors screen:

Home > Database provisioning > Manage database connectors

View Database Connectors							
Grid domain: CherylDomain							
You are viewing the database connectors in the current domain.							
Name	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security
MyDBConnector	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security
Create Connector		Done					

Step 2 Click the **View/Edit** link for the database connector that the database operation uses. The Database Connector Detail screen appears. The Allow/Disallow section is at the bottom of the screen.

Allow/Disallow Users					
The following users have access to create database operations for this database connector.					
Name	Type	Domain	Auth Service	Auth Service Type	Remove
Administrator	User	CherylDomain	DefaultAuthService	Grid	Remove
DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Remove
Add User		Add Group			

Step 3 To add a user to the list of users who can create a database operation that uses the database connection, click the **Add User** button. The Add Database Connector User screen

appears.

Add Database Connector User

All users
[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#)

In the Add column, place a check mark next to the names of the user(s) you want to allow to create database operations using database connector **MyDBConnector**

Add	Username	Authentication service	Authentication service type
<input type="checkbox"/>	Administrator	DefaultAuthService	Grid
<input type="checkbox"/>	MessagingUser	DefaultAuthService	Grid
<input type="checkbox"/>	Wilma	DefaultAuthService	Grid

Check All
Clear All

Submit
Cancel

Step 4 Click boxes in the Add column to select the users you want to add.

Step 5 Click **Submit** to save your changes. The system displays a list of the users you have added.

Allowing groups to create database operations

To allow a group to create a database operation that uses a designated database connection, the owner of the database connection must do the following:

Step 1 Navigate to the View Database Connectors screen:

Home > Database provisioning > Manage database connectors

View Database Connectors

Grid domain:

You are viewing the database connectors in the current domain.

Name	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security
MyDBConnector	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security

Create Connector
Done

- Step 2** Click the **View/Edit** link for the database connector that the database operation uses. The Database Connector Detail screen appears. The Allow/Disallow section is at the bottom of the screen.

Allow/Disallow Users					
The following users have access to create database operations for this database connector.					
Name	Type	Domain	Auth Service	Auth Service Type	Remove
Administrator	User	CherylDomain	DefaultAuthService	Grid	Remove
DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Remove
<input type="button" value="Add User"/> <input type="button" value="Add Group"/>					

- Step 3** To add a group to the list of groups who can create a database operation that uses the database connection, click the **Add Group** button. The Add Database Connector Group screen appears.

Add Database Connector Group			
All groups			
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z			
In the Add column, place a check mark next to the names of the group(s) you want to allow to create database operations using database connector MyDBConnector .			
Add	Name	Authentication service	Authentication service type
<input type="checkbox"/>	Administrators	DefaultAuthService	Grid
<input type="checkbox"/>	DataProviders	DefaultAuthService	Grid
<input type="checkbox"/>	DatabaseAdministrators	DefaultAuthService	Grid
<input type="checkbox"/>	DomainUsers	DefaultAuthService	Grid
<input type="checkbox"/>	MessagingUsers	DefaultAuthService	Grid
<input type="checkbox"/>	UserAdministrators	DefaultAuthService	Grid
<input type="checkbox"/>	admins	AvakiNIS	Nis
<input type="checkbox"/>	p4admins	AvakiNIS	Nis
<input type="checkbox"/>	testusers	AvakiNIS	Nis
<input type="checkbox"/>	users	AvakiNIS	Nis
<input type="button" value="Check All"/> <input type="button" value="Clear All"/>			
<input type="button" value="Submit"/> <input type="button" value="Cancel"/>			

- Step 4** Click boxes in the Add column to select the groups you want to add.

- Step 5** Click **Submit** to save your changes. The system displays a list of the groups you have added.

Preventing users from creating database operations

To prevent a user from creating a database operation that uses a designated database connection, do the following:

- Step 1** Navigate to the View Database Connectors screen:

Home > Database provisioning > Manage database connectors

View Database Connectors							
Grid domain: <input type="text" value="CherylDomain"/>							
You are viewing the database connectors in the current domain.							
Name	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security
MyDBConnector	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security
<input type="button" value="Create Connector"/>		<input type="button" value="Done"/>					

- Step 2** Click the **View/Edit** link for the database connector that the database operation uses. The Database Connector Detail screen appears. The Allow/Disallow section is at the bottom of the screen.

Allow/Disallow Users					
The following users have access to create database operations for this database connector.					
Name	Type	Domain	Auth Service	Auth Service Type	Remove
Administrator	User	CherylDomain	DefaultAuthService	Grid	Remove
DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Remove
<input type="button" value="Add User"/>		<input type="button" value="Add Group"/>			

- Step 3** To remove a user from the list of users who can create a database operation that uses the database connection, click **Remove**. The system removes the user from the list.

Preventing groups from creating database operations

To prevent a group from creating a database operation that uses a designated database connection, do the following:

Step 1 Navigate to the View Database Connectors screen:

Home > Database provisioning > Manage database connectors

View Database Connectors

Grid domain:

You are viewing the database connectors in the current domain.

Name	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security
MyDBConnector	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security

Step 2 Click the **View/Edit** link for the database connector that the database operation uses. The Database Connector Detail screen appears. The Allow/Disallow section is at the bottom of the screen.

Allow/Disallow Users

The following users have access to create database operations for this database connector.

Name	Type	Domain	Auth Service	Auth Service Type	Remove
Administrator	User	CherylDomain	DefaultAuthService	Grid	Remove
DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Remove

Step 3 To remove a group from the list of groups that can create a database operation that uses the database connection, click **Remove**. The system removes the group from the list.

Testing database connectors

Follow these steps to test whether a database connector is operational:

Step 1 Navigate to the View Database Connectors screen:

Home > Database provisioning > Manage database connectors

Step 2 Click the **Test** link beside the database connector you want to test. The system displays a message indicating whether the connector is operational.

Test Database Connector

The connection **MyDBConnector** is operational.

The following information was obtained from the database:

```

Database system type: Oracle
Database version: Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options
JDBC Driver type: Oracle JDBC driver
JDBC Driver version: 9.2.0.1.0

```

Managing SQL views

Follow these steps to manage any SQL views associated with a database connector:

Step 1 Navigate to the View Database Connectors screen:

Home > Database provisioning > Manage database connectors

View Database Connectors

Grid domain: ▼

You are viewing the database connectors in the current domain.

Name	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security
MyDBConnector	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security

- Step 2** Click the **SQL Views** link beside the database connector whose SQL views you want to manage. The Manage SQL Views screen appears, showing any SQL views that are based on the database connector.

Manage SQL Views							
Grid domain: CherylDomain ▾							
You are viewing the SQL views in the current domain.							
Name	Type	View/Edit	Remove	Attributes	Security	Categories	Schema
ScottStoreInfoSQLView	PROVISIONED	View/Edit	Remove	Attributes	Security	Categories	Schema
Provision SQL View		Done					

For information about managing SQL views, see [“Managing SQL views” on page 39](#).

Removing database connectors

Follow these steps to remove a database connector:

- Step 1** Navigate to the View Database Connectors screen:

Home > Database provisioning > Manage database connectors

View Database Connectors							
Grid domain: CherylDomain ▾							
You are viewing the database connectors in the current domain.							
Name	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security
MyDBConnector	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security
Create Connector		Done					

- Step 2** Click the **Remove** link beside the database connector you want to remove. The system displays a confirmation screen.
- Step 3** Click **OK** to confirm the operation. The system redisplay the View Database Connectors screen without the connector that you removed.

Managing database operations

This section describes how to perform various tasks with database operations, which extract data from a relational database and deliver it on demand to a view generator, a data service, or a generated SQL view.

In this section:

- [“Creating database operations,”](#) below
- [“Viewing and modifying database operations”](#) on page 29
- [“Viewing database operation details”](#) on page 29
- [“Managing database operation metadata”](#) on page 31
- [“Executing database operations”](#) on page 36
- [“Removing database operations”](#) on page 38

Creating database operations

To create a database operation, you must have the appropriate permissions for the related database connector, as follows:

- you must be the owner of a related database connector or the user who created it; or
- your database administrator must modify the permissions for the database connector that the database operation uses so that you are allowed to create a database operation (see [“Allowing users to create database operations”](#) on page 15).

Note To be capable of participating in a distributed transaction, a database operation must use a database connector that has been configured with an XA driver. The database connector configuration procedure, [“Creating database connectors”](#) on page 3, explains how to configure an XA driver.

To create a database operation, do the following:

- Step 1** Navigate to the Select Database Connector screen:

Home > Database provisioning > Create database operation

Select Database Connector	
Grid domain:	<input type="text" value="CherylDomain"/>
You have chosen to create a new database operation. Select the database connector to use.	
Select	Connector Name
<input type="radio"/>	MyDBCconnector
<input type="button" value="Continue"/> <input type="button" value="Cancel"/>	

Step 2 Select the database connector to use to create the database operation, then click **Continue**. The Create Database Operation screen appears.

Create Database Operation

Fill in the information about your database operation below. (If there's a file in your domain that contains your SQL statement, click **Browse** to import it.) If your SQL statement has one or more parameters, you will be asked to specify additional information about those parameters when you click **Continue**.

Database connector: MyDBConnector View Schema

Logical name:

Description (optional):

SQL statement:

Browse...

Updates database? Yes No

Cached data expiration: No caching
 Never expires
 Expires after seconds

Run database operation as: The user running the operation
 A specific user:
(e.g. `userName@authService.authServiceType.domain`)

Stored procedure? Yes No

Supports batch? Yes No

Continue Cancel

Step 3 Fill in the form:

- **Database connector:** Shows the name of the connector on which this database operation will be defined. Click the connector's name to view its details, or the **View**

Schema button to examine the catalogs, schemas, and tables in the connected database. For more information about viewing the database schema, see “[Viewing database schemas](#)” on page 10.

- Logical name: Enter a name for the database operation. **Note:** Do not include spaces in the name.
- Description (optional): Enter some descriptive information about this database operation.
- SQL statement: Enter the SQL statement that the database operation will execute. The SQL syntax must be valid for the underlying database, with one exception: Use a question mark (?) to specify parameters, as specified in the JDBC standard for specifying parameters in stored procedures. For details about using question marks to specify parameters, see the JDBC API documentation at:

<http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/getstart/PreparedStatement.html>

Alternatively, you can import the contents of a file that contains a SQL statement, as follows:

- Click the **Browse** button beneath the SQL statement box.
- In the Grid File Browser that appears, navigate to the file and select it.
- Click **Continue**. The contents of the file are inserted into the SQL statement box.

Note If you enter a SQL statement that involves an aggregate function such as SUM, you must use aliases for any column names to which the function refers. The aliases must follow these rules:

- All aliases must begin with a letter, an underscore (_), or a dollar sign (\$).
- All characters after the first character may be letters, numbers, an underscore (_), or a dollar sign (\$).
- All characters should be valid ASCII characters.
- An alias cannot be any of the following words:

break	enum	super
case	export	switch
catch	extends	this

class	finally	throw
comment	for	try
continue	function	typeof
const	if	var
debugger	import	void
default	in	while
delete	label	with
do	new	
else	return	

For example, in the following SQL statement, `dollarsum` is an alias for the column named `DOLLARS`.

```
SELECT sum(DOLLARS) dollarsum, CUSTOMERID FROM sales GROUP
BY CUSTOMERID ORDER BY CUSTOMERID
```

- **Updates database:** Select **Yes** if the database operation's SQL statement performs an update (such as `INSERT`, `DELETE`, or `UPDATE`); in this case, the database operation will not be executed when its schema is generated. Select **No** if the SQL statement does not perform an update, to enable the database operation to be executed when its schema is generated.
- **Cached data expiration:** Select one of the following to indicate whether the data is cached and, if so, when the data expires from the cache:
 - **No caching:** The data is not cached.
 - **Never expires:** The data never expires from the cache.
 - **Expires after *n* seconds:** Specify the interval (in seconds) before the data expires from the cache.
- **Run database operation as:** Specify which Avaki user the database operation will be run as. To run the operation as the current user, click **The user running the operation**.

To run the operation as a specific user: click **A specific user** and enter the qualified user name. Use the following format:

```
<user-name>@<authservice>.<authservicetype>.<domain>
```

For example:

```
wilma@DefaultAuthService.Grid.Bedrock
```

Alternatively, you can click the Browse link to open the User Browser window, then select a user and click **Continue**.

Note The Avaki web UI employs a special browsing feature when you're selecting the run-as user for a data service, database operation, or view. To make user browsing work properly, make sure the smooth scrolling option in your web browser is turned off. In Firefox or Internet Explorer, select Tools > (Internet) Options > Advanced and uncheck "Use smooth scrolling."

- Stored procedure? Select **Yes** if the SQL statement invokes a stored procedure in the underlying database, or select **No** if it doesn't. For the SQL syntax use the standard JDBC syntax for calling stored procedures via the CallableStatement object. For details about using CallableStatement, see the JDBC API documentation at:

<http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/getstart/callablestatement.html>

- Supports batch: Select **Yes** if the SQL statement is an update that can be used with JDBC batch mode.
- Click **Continue**.
- If the database operation requires parameters, a second Create Database Operation screen appears.

Create Database Operation

The SQL query for your database operation has one or more input parameters. For each parameter, specify its SQL type, then click **Submit** to create the database operation.

Query: select * from emp where ename = ?

Input Parameter 1: (Context: ...*ename* = ?)

If the database operation accepts input parameters or produces output parameters, use this option to specify the data type for each input and output. The table below shows the data types you can use:

BIGINT	LONGVARCHAR
BINARY	NUMERIC
BIT	ORACLE_CURSOR
BOOLEAN	OTHER
BLOB	REAL
CHAR	SMALLINT
CLOB	TIME
DATE	TIMESTAMP
DECIMAL	TINYINT
DOUBLE	VARBINARY
FLOAT	VARCHAR
INTEGER	Not a parameter
LONGVARBINARY	

If the input or output is not a parameter but is part of a known operator, select “Not a parameter” as the data type.

Click **Submit** to update the database operation.

- Step 4** Optional. To test whether a database operation is working properly, click the **Test** button. The system displays the Execute Database Operation page. For information about executing a database operation, see [“Executing database operations” on page 36](#).

Execute Database Operation

Specify the values of the input parameters for database operation **CherylDomain.MyDBConnector.MyDBOperation**, and then click **Submit** to execute it. Alternatively, you may [browse](#) for an XML input file.

SQL statement: select * from emp where ename = ?

Parameter 1: VARCHAR

Generate schema and suppress output:

For information about configuring advanced database operation settings, see [Appendix A, “Advanced database management”](#).

Viewing and modifying database operations

To view or modify a list of database operations in the current grid domain, navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations								
Grid domain: <input type="text" value="CherylDomain"/>								
You are viewing the database operations in the current domain.								
Sort by <input checked="" type="radio"/> Name <input type="radio"/> Connector								
Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
<input type="button" value="Create Operation"/>		<input type="button" value="Done"/>						

The system displays a list of the database operations in the current Avaki domain. By default, the list is sorted by database operation names. To sort the list by database connector names, select Connector.

Viewing database operation details

To view details about a database operation, do the following:

- Step 1** Navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations								
Grid domain: <input type="text" value="CherylDomain"/>								
You are viewing the database operations in the current domain.								
Sort by <input checked="" type="radio"/> Name <input type="radio"/> Connector								
Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
<input type="button" value="Create Operation"/>		<input type="button" value="Done"/>						

- Step 2** Click the **View/Edit** link beside the name of the database operation whose details you want to view or modify. The Update Database Operation screen appears, showing the details for the database operation.

Update Database Operation

To update the selected database operation, please modify the information below, and click **Continue**. (If there's a file in your domain that contains your SQL statement, click **Browse** to import it.) If the SQL statement has one or more parameters, you will then be given a chance to update their types.

Database connector: MyDBConnector

Logical name: MyDBOperation

Description:

Metadata: [Click here](#) to generate or view the schema for this database operation, generate or remove it as a virtual SQL view, or browse its dependencies.

SQL statement:

SQL parameters: in:VARCHAR

Updates database? Yes No

Cached data expiration: No Caching
 Never Expires
 Expires after seconds

Run database operation as: The user running the operation
 A specific user:
(e.g. *userName@authService.authServiceType.domain*)

Stored procedure: Yes No

Supports batch: Yes No

Step 3 Modify the fields if desired, then click **Continue**. A confirmation screen appears when the update is complete.

Managing database operation metadata

You can generate, view, or regenerate a database operation's schema (and view it if it has been generated), browse a database operation's dependencies, and expose the results of a database operation as a SQL view.

Generating a database operation's schema

To generate schema information, do the following:

- Step 1** Navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations

Grid domain:

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

- Step 2** Click the **Metadata** link beside the database operation whose metadata you want to manage. The Manage Metadata screen appears.

Manage Metadata

Manage the metadata for database operation **CherylDomain.MyDBConnector.MyDBOperation**: generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.

Schema: [Generate](#) [View](#)

Dependencies: [Browse](#)

SQL View

You must generate this database operation's schema before you can expose it as a SQL view.

- Step 3** Optional. To generate schema information, click **Generate**. You must generate this database operation's schema if you want to expose it as a SQL view.

Step 4 Optional. To view generated schema information, click **View**. The Avaki Schema Viewer appears, showing the table's schema.

CherylDomain.MyDBConnector.MyDBOperation									
Columns:									
Index	Name	Type	Precision	Scale	Display Size	Allows Nulls	Auto Increment	Case Sensitive	Read-Only
1	EMPNO	NUMERIC	4	0	22	false	false	false	false
2	ENAME	VARCHAR	10	0	10	true	false	true	false
3	JOB	VARCHAR	9	0	9	true	false	true	false
4	MGR	NUMERIC	4	0	22	true	false	false	false
5	HIREDATE	DATE	0	0	7	true	false	false	false
6	SAL	NUMERIC	7	2	22	true	false	false	false
7	COMM	NUMERIC	7	2	22	true	false	false	false
8	DEPTNO	NUMERIC	2	0	22	true	false	false	false

Step 5 Optional. To regenerate schema information, click **Refresh**. You should regenerate the schema information if the structure of the database operation's resultset changes. The structure will change if either of the following conditions occurs:

- the database operation's SQL statements changes; or
- the schema of the underlying database table changes.

If the database operation's resultset structure changes, you should propagate the change to any data services or SQL views that depend on the database operation. For information about displaying a list of services or views that depend on the database operation, see "[Viewing database operation dependencies](#)," below.

Viewing database operation dependencies

You can view a list of the data services, SQL views and view generators (if any) that depend on the database operation. To view these dependencies, do the following:

Step 1 Navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations

Grid domain: CherylDomain ▾

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

Create Operation
Done

Step 2 Click the **Metadata** link beside the database operation whose metadata you want to manage. The Manage Metadata screen appears.

Manage Metadata

Manage the metadata for database operation **CherylDomain.MyDBConnector.MyDBOperation**: generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.

Schema: [Generate](#) [View](#)

Dependencies: [Browse](#)

Done

Step 3 To view database operation dependencies, click **Browse**.

The system lists the data services, SQL views and view generators that depend on the database operation.

Dependencies for CherylDomain.MyDBConnector.MyDBOperation

Operation	Dependency	Dependency Type
Data Service	CherylDomain.MyDataService	Input from CherylDomain.MyDBConnector.MyDBOperation

Done

For details about creating data services, see [Chapter 2, “Basic data integration” on page 49](#). For details about creating view generators, see [Chapter 6, “Managing views”](#)

on page 217. For details about exposing database operation results as a SQL view, see below.

Exposing database operation results as a SQL view

You can expose the database operation results as a SQL view, so that the database operation resultset can be operated on as a table via JDBC or a virtual database operation. Such a SQL view is static in two ways: first, if the database operation's schema changes, you must regenerate the SQL view accordingly; second, the regeneration of a database operation's SQL view can only be done as two separate steps—removing the old SQL view, followed by generating a new one. Here's how you manage a database operation's SQL view:

Step 1 Navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations

Grid domain:

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

Step 2 Click the **Metadata** link beside the database operation whose metadata you want to manage. The Manage Metadata screen appears.

- If there's not yet a SQL view representing this database operation (but the operation's schema has been generated), the screen looks like this:

Manage Metadata
<p>Manage the metadata for database operation CherylDomain.MyDBConnector.MyDBOperation: generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.</p> <p>Schema: Refresh View</p> <p>Dependencies: Browse</p> <p><input type="button" value="Done"/></p>

SQL View
<p>Logical name: <input type="text"/> Generate Remove</p>

In the Logical Name box, enter a logical name for the SQL view to be generated. Do not include any spaces in the name. Click the **Generate** link.

- If there is a SQL view representing this database operation, the screen displays the logical name of the SQL view:

Manage Metadata
<p>Manage the metadata for database operation CherylDomain.MyDBConnector.MyDBOperation: generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.</p> <p>Schema: Refresh View</p> <p>Dependencies: Browse</p> <p><input type="button" value="Done"/></p>

SQL View
<p>Logical name: MyDBOperationTable Generate Remove</p>

Removing SQL views

To remove a SQL view generated from a database operation, do the following:

- Step 1** Navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations

Grid domain:

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

- Click the **Metadata** link beside the database operation whose metadata you want to manage. The Manage Metadata screen appears. The screen displays the logical name of the SQL view:

Manage Metadata

Manage the metadata for database operation **CherylDomain.MyDBConnector.MyDBOperation**: generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.

Schema: [Refresh](#) [View](#)

Dependencies: [Browse](#)

SQL View

Logical name: MyDBOperationTable [Generate](#) [Remove](#)

- Step 2** Click the **Remove** link beside the logical name of the SQL view that you want to delete.

Executing database operations

You can display the results of a database operation to make sure that the operation is set up correctly and that the connections are healthy.

To execute a database operation, you must have the appropriate permissions for the operation, as follows:

- you must be the user who created the related database operation; or

- your database administrator must modify the access control list (ACL) for the database operation so that you are allowed to execute it. The database administrator must give you permission to read and execute the database operation. For details about configuring ACLs, see the *Sybase Avaki EII Administration Guide*.

Follow these steps to execute a database operation:

Step 1 Navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations

Grid domain:

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

Step 2 Click the **Execute** link beside the name of the database operation you want to execute. The Execute Database Operation page appears.

Execute Database Operation

Specify the values of the input parameters for database operation **CherylDomain.MyDBConnector.MyDBOperation**, and then click **Submit** to execute it. Alternatively, you may [browse](#) for an XML input file.

SQL statement: select * from emp where ename = ?

Parameter 1: VARCHAR

Generate schema and suppress output:

Step 3 Optional. If your query requires parameters, enter the parameters, or click the **browse** link and navigate to an XML input file that contains the parameters. The XML input file must conform to the rowset-specific schema described in [Appendix D, “Avaki rowset XML”](#) (see “[Rowset-specific schema](#)” on page 280). For each value in the input file, you must define a matching SQL parameter in the database operation. In the schema on [page 280](#), for example, you must define parameters for the company name, for each part of the company’s address, and for the ID number.

- Step 4** Optional. Select the **Generate schema and suppress output** option to generate schema for the database operation. The schema describes the structure of the output of a database operation.
- Step 5** Click **Submit**. A confirmation screen appears. If you chose to generate schema in the previous step, you can click **View Schema** to view the database operation's schema. If you chose not to generate schema, the results of the query appear in XML format.

Calling a database operation. You can use ODBC, JDBC, or SOAP procedure calls to call a database operation in the data catalog. For more information, see the *Sybase Avaki EII API Guide*.

Removing database operations

Follow these steps to remove a database operation:

- Step 1** Navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations								
Grid domain: <input type="text" value="CherylDomain"/>								
You are viewing the database operations in the current domain.								
Sort by <input checked="" type="radio"/> Name <input type="radio"/> Connector								
Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
<input type="button" value="Create Operation"/>		<input type="button" value="Done"/>						

- Step 2** Click the **Remove** link beside the database operation you want to remove. The system displays a confirmation screen.
- Step 3** Click **OK** to confirm the operation. The system redisplayes the View Database Operations screen without the operation that you removed.

Managing SQL views

A SQL view is data that can be accessed using standard SQL statements by connecting to Avaki with ODBC or JDBC, or via an Avaki virtual database operation. It is a virtual table and can be made available to applications that issue SQL statements, such as certain business intelligence and reporting applications, or to business users issuing ad-hoc queries.

A SQL view or set of SQL views provides a distributed query capability, where Avaki optimizes execution by delegating processing to the database management systems where appropriate. The advantage of a SQL view is its flexibility in serving applications and ad-hoc queries. The disadvantage is that this flexibility sometimes conflicts with restrictions data owners need to place on how production databases can be used.

Note When you provision a table into Avaki from a relational DBMS, Avaki uses a default set of mappings between the SQL data types used in the DBMS and the data types used by Avaki's query engine. If the default data type mappings yield unsatisfactory results, you can override them for a particular database connection. For instructions on overriding the default data type mappings, see the *Sybase Avaki EII Command Reference*.

The following sections describe how to provision a table as a SQL view, and how to view and modify SQL views:

- [“Provisioning SQL views,”](#) below
- [“Viewing SQL views”](#) on page 43
- [“Modifying SQL views”](#) on page 43

Provisioning SQL views

When you provision a SQL view, you make a table from a connected database visible in Avaki.

To provision a database table as a SQL view, do the following:

Step 1 Navigate to the Provision SQL View screen:

Home > Database provisioning > Provision SQL view

Provision SQL View

Select (or enter the qualified names of) a database connector and one of its tables below. Then give the table an Avaki name and, optionally, a description. Click **Continue** to provision the table.

Database connector:

Table to be provisioned:

Logical name:

Description (optional):

Step 2 Click the **Browse** button beside the Database connector field. The Database Connection Selector screen appears.

Database Connection Selector

Select the radio button for the desired database connector and click **Continue**.

Grid domain:

MyDBConnector

Step 3 Select the database connector that connects to the relational database that contains the table to provision and click **Continue**.

- Step 4** Click the **Browse** button beside the Table to be provisioned field. The pop-up Database Table Selector screen appears. The Database Table Selector is a version of the Database Table Browser (see [“Viewing database schemas” on page 10](#)) that allows you not only to view the schema of the connected database but also to select one of its tables for provisioning.

Database Table Selector

Examine the schemas and tables in the connected database. Navigate to the table you want, select it, and click **Continue**.

schema: ▼

Tables:

- PUBLIC.ALL.ALL TABLES
- PUBLIC.ALL.APPLY
- PUBLIC.ALL.APPLY.CONFLICT COLUMNS
- PUBLIC.ALL.APPLY.DML HANDLERS
- PUBLIC.ALL.APPLY.ENQUEUE
- PUBLIC.ALL.APPLY.ERROR

- Step 5** As with the Database Table Browser, you can narrow down (or populate) the list of tables on the left by choosing a catalog or schema (or both) from the available pull-down menus, and you can view the schema of a particular table by clicking on its name in that list:

Database Table Selector

Examine the schemas and tables in the connected database. Navigate to the table you want, select it, and click **Continue**.

schema:

Tables:

- ANSWER
- BIN\$1hJVhzNORzaRSIpm2wPd6w==\$0
- BIN\$32VWn0F6QZw5Ww/c1ZPAGUw==\$0
- BIN\$A7ck/lpsQ5y1rYyXY8H7zQ==\$0
- BIN\$CnxeiJQZTFOLWb6QfhlGlw==\$0
- BIN\$LhQ+yw7@T0Cnij/G3q3xCg==\$0
- BIN\$Q36+Pdk9TsesNtrSBACYA==\$0
- BIN\$Y3+wAEl8TKCczTrNKv3uMA==\$0
- BIN\$YVQFZgnARiazv5I+CjzZow==\$0
- BIN\$aT5CdJzpTPCS9D0tc55fzw==\$0
- BIN\$f0KxjX19Qp@BBILv6Jjq3Q==\$0
- BIN\$hvMHfutQH+KTIsvAeWzZA==\$0
- BIN\$hc/aYmw5QXyOdgvaeh8elg==\$0
- BIN\$hd8HypxSc+SxgncAsfCw==\$0
- BIN\$o4+Q/RRmThvl8hevHFMUUQ==\$0
- BIN\$APhUH/9ShuUINDpranWwWQ==\$0
- BIN\$vozfTjMfQJG5vRk7RtoH5g==\$0
- BIN\$wwwrrBr6T5iSdajr3K9mqw==\$0
- BIRTHDAYS
- BONUS
- CATEGORIESTESTDB
- DEPT
- DOUBLETEST
- EMP
- EMP2

Table name: **SCOTT.EMP**

Column Metadata

Name	Type	Size	Scale	Nullable
EMPNO	DECIMAL	4	0	false
ENAME	VARCHAR	10	None	true
JOB	VARCHAR	9	None	true
MGR	DECIMAL	4	0	true
HIREDATE	TIMESTAMP	7	None	true
SAL	DECIMAL	7	2	true
COMM	DECIMAL	7	2	true
DEPTNO	DECIMAL	2	0	true

Primary Key Columns

Column	Sequence
EMPNO	1

Tables Referenced by Foreign Keys

Primary Key Table	Foreign Key Name	Foreign Key Column (SeqNo)	Update Rule	Delete Rule
DEPT	FK_DEPTNO	DEPTNO(1)	None	Restrict

- Step 6** The Database Table Selector also offers a radio button next to each table visible in the list on the left. To select a table for provisioning, click the radio button beside its name and click **Continue** to close the Database Table Selector window.
- Step 7** In the Logical name box, enter a name for the SQL view.
- Step 8** Optional. In the Description box, enter a description for the SQL view.
- Step 9** Click **Continue**. The system displays a confirmation screen.

Viewing SQL views

- Step 1** To list and display information about the SQL views in your Avaki domain, navigate to the Manage SQL Views screen:

Home > Database provisioning > Manage SQL views

Manage SQL Views

Grid domain:

You are viewing the SQL views in the current domain.

Name	Type	View/Edit	Remove	Attributes	Security	Categories	Schema
ScottStoreInfoSQLView	PROVISIONED	View/Edit	Remove	Attributes	Security	Categories	Schema

Modifying SQL views

- Step 1** To modify a SQL view, navigate to the Manage SQL Views screen:

Home > Database provisioning > Manage SQL views

Manage SQL Views

Grid domain:

You are viewing the SQL views in the current domain.

Name	Type	View/Edit	Remove	Attributes	Security	Categories	Schema
ScottStoreInfoSQLView	PROVISIONED	View/Edit	Remove	Attributes	Security	Categories	Schema

- Step 2** Click on the name of the SQL view to modify.

Step 3 The Provision SQL View screen appears.

Provision SQL View

Click **Continue** to change this table's description, or use the links in the **Activities** section to **Remove** the table or view and edit its **Attributes**, **Security** settings or **Categories**.

Database connector: CherylDomain.MyDBConnector
 Table to be provisioned: SCOTT.STORE_INFO
 Logical name: ScottStoreInfoSQLView
 Description (optional):

Activities

[Remove](#)
[Attributes](#)
[Security](#)
[Categories](#)

Step 4 Modify the description as needed, then click **Continue** to save your changes.

Removing SQL views

Follow these steps to remove a SQL view:

Step 1 Navigate to the Manage SQL Views screen:

Home > Database provisioning > Manage SQL views

Manage SQL Views

Grid domain: ▼

You are viewing the SQL views in the current domain.

Name	Type	View/Edit	Remove	Attributes	Security	Categories	Schema
ScottStoreInfoSQLView	PROVISIONED	View/Edit	Remove	Attributes	Security	Categories	Schema

Step 2 Click the **Remove** link beside the database connector you want to remove. The system displays a confirmation screen.

Step 3 Click **OK** to confirm the operation. The system redisplay the Manage SQL Views screen without the SQL view that you removed.

Configuring SQL view attributes

Each SQL view has attributes that store information such as the time at which the SQL view was created and the name of the user who owns the service.

To view the attributes for a SQL view, do the following:

- Step 1** Navigate to the Manage SQL Views screen:

Home > Data integration > Manage SQL views

Manage SQL Views

Grid domain: CherylDomain ▼

You are viewing the SQL views in the current domain.

Name	Type	View/Edit	Remove	Attributes	Security	Categories	Schema
ScottStoreInfoSQLView	PROVISIONED	View/Edit	Remove	Attributes	Security	Categories	Schema

Provision SQL View
Done

- Step 2** Click the **Attributes** link. The View Attributes page appears, showing the attributes for the SQL view.

System Attributes		
Name	Type	Value
system/ObjectType	string	SQL View
system/SqlViewType	string	PROMISIONED
system/provisionedSqlViewSourceType	string	TABLE
system/provisionedSqlViewSourceTable	string	STORE_INFO
system/provisionedSqlViewSourceSchema	string	SCOTT
system/SqlViewDataSourceName	string	CherylDomain.MyDB.Connector
system/QualifiedName	string	CherylDomain.MyDB.Connector.ScottStoreInfoSQLView
system/ObjectHostName	string	magadieu.avaki.local:1299
system/ObjectHostJndiPort	string	1299
system/References	integer	1
system/ObjectLoid	string	loid://1/100b9ba0d4/100b9ba41d/100b9ba69a
system/ObjectType	string	SQL View
system/ChangeTime	timestamp	2004-12-09 19:53:47.297
system/ModificationTime	timestamp	2004-12-09 19:53:49.551
system/Owner	string	/System/Domains/CherylDomain/Services/AuthServices/Grnd/DefaultAuthService/Users/Administrator

User-defined Attributes		
A user-defined attribute may be one of the following types:		
Type	Description	Format
String	Text	Any characters
Integer	Any whole quantity	Any integer
Float	A numeric value that can be fractional or very large	Any numeric value
Date	Year, month, and date	yyyy-mm-dd
Time	Hour, minute and second that an event occurs	hh:mm:ss
Timestamp	A precise time and date	yyyy-mm-dd hh:mm:ss.ffffff
This item currently has no user attributes.		
To add a new attribute, specify a name, type, and value in the fields below.		
Name: <input type="text"/>	Type: <input type="text" value="String"/>	Value: <input type="text"/>

For details about creating, modifying, and deleting attributes, see the *Sybase Avaki EII Administration Guide*.

Managing SQL view ACLs

An access control list (ACL) determines which grid users are allowed to read and manipulate each SQL view.

To view the ACL for a SQL view, do the following:

- Step 1** Navigate to the Manage SQL Views screen:

Home > Data integration > Manage SQL views

Manage SQL Views

Grid domain: CherylDomain

You are viewing the SQL views in the current domain.

Name	Type	View/Edit	Remove	Attributes	Security	Categories	Schema
ScottStoreInfoSQLView	PROVISIONED	View/Edit	Remove	Attributes	Security	Categories	Schema

Provision SQL View
Done

- Step 2** Click the **Security** link. The View Security Information appears, showing the users and groups that have been added to the ACL for the SQL view.

View Security Information

You are viewing security information for the following object:
/System/Domains/CherylDomain/Services/DatabaseServices/MyDBConnector/ProvisionedSQLViews/ScottStoreInfoSQLView

The current owner of the object is the User **Administrator** in domain **CherylDomain** and the Grid authentication service **DefaultAuthService**.

The following users and groups have been added to the Access Control List (ACL) for this object. To modify a user or group's permissions, place a check mark next to the user or group and click **Edit All Checked**. To add a user who is in the current grid domain, click **Add User to ACL**. To add a group that is in the current domain, click **Add Group to ACL**. To add a user or group that is in a connected domain, click **Add Via Interconnect ID**.

Select	Name	Type	Domain	Auth Service	Auth Service Type	Read	Write	Execute	Delete
<input type="checkbox"/>	DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Allow	Unset	Unset	Unset
<input type="checkbox"/>	Administrator	User	CherylDomain	DefaultAuthService	Grid	Allow	Allow	Allow	Allow

Check All
Clear All

Manage Access Control Lists

[Edit All Checked](#) Configure permissions for each user or group in the ACL for an object

[Add User to ACL](#) Add a user to the ACL for an object

[Add Group to ACL](#) Add a group to the ACL for an object

[Add Via Interconnect ID](#) Upload an interconnect ID to add a user or group to the ACL for an object

Done

For details about viewing and modifying ACLs, see the *Sybase Avaki EII Administration Guide*.

Managing SQL view categories

Categories let you classify and organize the contents of your data catalog. To assign a SQL view to a category, do the following:

- Step 1** Navigate to the Manage SQL Views screen:

Home > Data integration > Manage SQL views

Manage SQL Views

Grid domain: ▼

You are viewing the SQL views in the current domain.

Name	Type	View/Edit	Remove	Attributes	Security	Categories	Schema
ScottStoreInfoSQLView	PROVISIONED	View/Edit	Remove	Attributes	Security	Categories	Schema

Click the **Categories** link. The View Categories screen appears.

View Categories

/System/Domains/CherylDomain/Services/DatabaseServices/MyDBConnector/ProvisionedSQLViews/ScottStoreInfoSQLView does not belong to any categories.

Add this object to a category:

For details about adding objects to categories and removing them from categories, see the *Sybase Avaki EII Administration Guide*.

Basic data integration

There are two mechanisms that you can use, separately or together, to integrate data from multiple sources within Avaki: Avaki data services, and the Avaki virtual database.

The Avaki virtual database lets you create a special kind of database operation, the *virtual database operation*, which operates on Avaki SQL views. Once database tables (and/or database operation results or data service results—see below) are provisioned into Avaki as SQL views, virtual database operations allow you to use SQL to integrate those tables and results. And you needn't stop there: like other database operations, virtual database operations too can be exposed as SQL views.

Avaki data services go beyond SQL to let you integrate data from multiple heterogeneous distributed data sources, including files, relational data, XML data, and application data. Data services also give architects and developers maximum flexibility in packaging and reusing data integration logic. Unlike database operations, which always return relational data (or Avaki's XML representation thereof), data services can produce data in any format desired (albeit perhaps with some programming required on the part of your developers). Avaki Studio is a graphical tool that goes a long way toward unlocking the power of data services for you without requiring you to be a programmer; it is discussed more fully in *Data Integration with Sybase Avaki Studio*.

Data services and the virtual database can work together, too. As mentioned above, data services that return relational data can be exposed as SQL views (just like database operations); and virtual database operations, like all database operations, can be used as data service inputs.

This chapter discusses the Avaki virtual database and introduces data services. It covers the following topics:

- [“Virtual database operations,”](#) below
- [“Managing virtual database services”](#) on page 64
- [“Data services overview”](#) on page 75
- [“Creating data services”](#) on page 81
- [“Importing data service descriptors”](#) on page 92
- [“Viewing a list of data services”](#) on page 94
- [“Modifying data services”](#) on page 94
- [“Managing data service metadata”](#) on page 97
- [“Testing data services”](#) on page 102
- [“Removing data services”](#) on page 103

Virtual database operations

A virtual database operation extracts data from SQL views in the virtual database and delivers it on demand to a view generator or a data service.

Creating virtual database operations

To create a virtual database operation, you must have the appropriate permissions for any related database connectors or SQL views as well as the virtual database service, as follows:

- Your database administrator must modify the permissions for any database connectors or SQL views that the virtual database operation uses so that you have read, write, and execute permission for the database connectors and SQL views (for information about setting access permissions, see the *Sybase Avaki EII Administration Guide*).
- In addition, your database administrator must modify the permissions for the virtual database service so that you are allowed to create virtual database operations on the service (see [“Allowing users to create virtual database operations”](#) on page 66.)

To create a virtual database operation, do the following:

Step 1 Navigate to the Create Virtual Database Operation screen:

Home > Data integration > Create virtual database operation

Create Virtual Database Operation

Fill in the information about your virtual database operation below. (If there's a file in your domain that contains your SQL statement, click **Browse** to import it.) If your SQL statement has one or more parameters, you will be asked to specify additional information about those parameters when you click **Continue**.

Database schema:

Server:

Logical name:

Description (optional):

SQL statement:

Cached data expiration: No caching
 Never expires
 Expires after seconds

Run database operation as: The user running the operation
 A specific user:
(e.g. userName@authService.authServiceType.domain)

Step 2 Fill in the form:

- Database schema (optional): Click the **View Schema** button to use the pop-up Database Table Browser to examine the tables defined in the Avaki virtual database. In JDBC terms, each Avaki domain is a “catalog,” which contains a variable number of schemas, which in turn may contain table definitions. There are four kinds of schemas in each domain:
 - DATASERVICE: Contains all SQL views generated from data services.
 - <Database-Connector-Name>: Contains all SQL views provisioned from the database to which the given database connector connects or generated from database operations on this database connector.
 - VIRTUALDB: Contains all SQL views generated from virtual database operations.
 - <Metadata-Model-Name>: Contains all the mapped tables deployed from the given metadata model. Note that if a catalog other than the Avaki domain name was specified on deployment of a metadata model, that metadata model/schema and its tables will belong to the specified catalog, not to your domain’s catalog. For more on metadata models, see *Data Integration with Sybase Avaki Studio*.

For more information about using the Database Table Browser, see “[Viewing database schemas](#)” on page 10.

- Server: From the pull-down list, select the server on which to create the virtual database operation.
- Logical name: Enter a name for the virtual database operation. **Note:** Do not include a space in the name.
- Description (optional): Enter some descriptive information about this virtual database operation.
- SQL statement: Enter the SQL statement that the virtual database operation will execute. The SQL syntax must be valid for the underlying database, with one exception: Use a question mark (?) to specify parameters, as specified in the JDBC standard for specifying parameters in stored procedures. For details about using question marks to specify parameters, see the JDBC API documentation at:

<http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/getstart/PreparedStatement.html>

Alternatively, you can import the contents of a file that contains a SQL statement, as follows:

- Click the **Browse** button beneath the SQL statement box.
- In the Grid File Browser that appears, navigate to the file and select it.
- Click **Continue**. The contents of the file are inserted into the SQL statement box.

Note If you enter a SQL statement that involves an aggregate function such as SUM, you must use an alias for any column names to which the function refers. The alias must follow these rules:

- All aliases must begin with a letter, an underscore (_), or a dollar sign (\$).
- All characters after the first character may be letters, numbers, an underscore (_), or a dollar sign (\$).
- All characters should be valid ASCII characters.
- An alias cannot be the same name as any of the following words:

break	enum	super
case	export	switch
catch	extends	this
class	finally	throw
comment	for	try
continue	function	typeof
const	if	var
debugger	import	void
default	in	while
delete	label	with
do	new	
else	return	

For example, in the following SQL statement, `dollarsum` is an alias for the column named `DOLLARS`:

```
SELECT sum(DOLLARS) dollarsum, CUSTOMERID FROM sales GROUP
BY CUSTOMERID ORDER BY CUSTOMERID
```

- **Cached data expiration:** Select one of the following to indicate whether the data is cached and, if so, when the data expires from the cache:
 - **No caching:** The data is not cached.
 - **Never expires:** The data never expires from the cache.
 - **Expires after *n* seconds:** Specify the interval (in seconds) before the data expires from the cache.
- **Run database operation as:** Specify which user the virtual database operation will be run as. To run the operation as the current user, click **The user running the operation**. To run the operation as a specific user, click **A specific user** and enter the qualified user name. Use the following format:

```
<user-name>@<authservice>.<authservicetype>.<domain>
```

For example:

```
wilma@DefaultAuthService.Grid.Bedrock
```

Note The Avaki web UI employs a special browsing feature when you're selecting the run-as user for a data service, database operation, or view. To make user browsing work properly, make sure the smooth scrolling option in your web browser is turned off. In Firefox or Internet Explorer, select Tools > (Internet) Options > Advanced and uncheck "Use smooth scrolling."

- Click **Continue**.
- If the database operation requires parameters, another Create Virtual Database Operation screen appears.

Create Virtual Database Operation

The SQL query for your virtual database operation has one or more input parameters. For each parameter, specify its SQL type, then click **Submit** to create the virtual database operation.

Query: `select * from emp where ename = ?`

Input Parameter 1: (Context: `...ename = ?`)

If the virtual database operation accepts input parameters or produces output parameters, use this option to specify the data type for each input and output. The table below shows the data types you can use:

BIGINT	LONGVARCHAR
BINARY	NUMERIC
BIT	ORACLE_CURSOR
BOOLEAN	OTHER
BLOB	REAL
CHAR	SMALLINT
CLOB	TIME
DATE	TIMESTAMP
DECIMAL	TINYINT
DOUBLE	VARBINARY
FLOAT	VARCHAR
INTEGER	Not a parameter
LONGVARBINARY	

If the input or output is a not a parameter but is part of a known operator, select “Not a parameter” as the data type.

Click **Submit** to update the database operation

- Step 3** Optional. To test whether a virtual database operation is working properly, click the **Test** button. The system displays the Execute Virtual Database Operation page. For information about executing a virtual database operation, see [“Executing virtual database operations” on page 62](#).

Execute Virtual Database Operation

Specify the values of the input parameters for virtual database operation **CherylDomain.VirtualDb.MyVirtualDBOperation**, and then click **Submit** to execute it.

SQL statement: select * from emp where ename = ?

Parameter 1: VARCHAR

Generate schema and suppress output:

Viewing and modifying virtual database operations

To view a list of virtual database operations in the current grid domain,

- Step 1** Navigate to the View Virtual Database Operations screen:

Home > Data integration > Manage virtual database operations

View Virtual Database Operations

Grid domain:

You are viewing the virtual database operations in the current domain.

Name	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyVirtualDBOperation	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

The system displays a list of the virtual database operations in the current grid domain.

- Step 2** To view details about a virtual database operation, click the **View/Edit** link beside the name of the virtual database operation whose details you want to view. The Update Virtual Database Operation screen appears, showing details on the database operation.

Update Virtual Database Operation

Fill in the information about your virtual database operation below. (If there's a file in your domain that contains your SQL statement, click **Browse** to import it.) If your SQL statement has one or more parameters, you will be asked to specify additional information about those parameters when you click **Continue**.

Database schema:

Server: MAGADIEUXP.sybase.com

Logical name: MyVirtualIDBOP

Description (optional):

Metadata: [Click here](#) to generate or view the schema for this virtual database operation, generate or remove it as a virtual SQL view, or browse its dependencies.

SQL statement:

Cached data expiration: No caching
 Never expires
 Expires after seconds

Run database operation as: The user running the operation
 A specific user: [Browse](#)
(e.g. userName@authService.authServiceType.domain)

- Step 3** Optional. Modify the fields as desired, then click **Continue**. A confirmation screen appears when the update is complete.

Managing metadata for virtual database operations

You can generate or regenerate a virtual database operation's schema (and view it if it has been generated), browse the virtual database operation's dependencies, and expose the results of the virtual database operation as a SQL view.

Generating a virtual database operation's schema

To generate schema information, do the following:

- Step 1** Navigate to the View Virtual Database Operations screen:

Home > Data integration > Manage virtual database operations

View Virtual Database Operations

Grid domain:

You are viewing the virtual database operations in the current domain.

Name	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyVirtualDBOperation	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

- Step 2** Click the **Metadata** link beside the virtual database operation whose metadata you want to manage. The Manage Metadata screen appears.

Manage Metadata

Manage the metadata for virtual database operation **CherylDomain.VirtualDb.MyVirtualDBOperation**: generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.

Schema: [Generate](#) [View](#)

Dependencies: [Browse](#)

- Step 3** To generate schema information, click **Generate**.

- Step 4** Optional. To view generated schema information, click **View**. The Avaki Schema Viewer appears, showing the table's schema.

CherylDomain.VirtualDb.MyVirtualDBOperation

Columns:

Index	Name	Type	Precision	Scale	Display Size	Allows Nulls	Auto Increment	Case Sensitive	Read-Only
1	EMPNO	DECIMAL	4	4	22	true	false	false	false
2	ENAME	VARCHAR	10	10	128	true	false	true	false
3	JOB	VARCHAR	9	9	128	true	false	true	false
4	MGR	DECIMAL	4	4	22	true	false	false	false
5	HIREDATE	TIMESTAMP	0	0	29	true	false	false	false
6	SAL	DECIMAL	7	7	22	true	false	false	false
7	COMM	DECIMAL	7	7	22	true	false	false	false
8	DEPTNO	DECIMAL	2	2	22	true	false	false	false
9	Employee	VARCHAR	0	0	128	true	false	true	false

Step 5 Optional. To regenerate schema information, click **Refresh**. You should regenerate the schema information if the structure of the virtual database operation’s resultset changes. The structure will change if either of the following conditions occurs:

- the virtual database operation’s SQL statements changes; or
- the SQL view on which the database operation operates changes.

If the virtual database operation’s resultset structure changes, you should propagate the change to any data services or SQL views that depend on the virtual database operation. For information about displaying a list of services or views that depend on the virtual database operation, see “[Viewing virtual database operation dependencies](#),” below.

Viewing virtual database operation dependencies

You can view a list of the data services, SQL views, and view generators (if any) that depend on the virtual database operation. To view these dependencies, do the following:

Step 1 Navigate to the View Virtual Database Operations screen:

Home > Data integration > Manage virtual database operations

View Virtual Database Operations

Grid domain:

You are viewing the virtual database operations in the current domain.

Name	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyVirtualDBOperation	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

Step 2 Click the **Metadata** link beside the virtual database operation that is used as an input source. The Manage Metadata screen appears.

Manage Metadata

Manage the metadata for virtual database operation **CherylDomain.VirtualDb.MyVirtualDBOperation**: generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.

Schema: [Generate](#) [View](#)

Dependencies: [Browse](#)

Step 3 To view database operation dependencies, click **Browse**.

- Step 4** The system lists the data services, SQL views and view generators that depend on the virtual database operation.

Dependencies for CherylDomain.VirtualDb.MyVirtualDBOperation		
Operation	Dependency	Dependency Type
Data Service	CherylDomain.MyDataService	Input from CherylDomain.VirtualDb.MyVirtualDBOperation

For details about creating data services, see [Chapter 2, “Basic data integration” on page 49](#). For details about creating view generators, see [Chapter 6, “Managing views” on page 217](#). For details about exposing database operation results as a SQL view, see below.

Exposing virtual database operation results as a SQL view

You can expose the virtual database operation results as a SQL view, so that the virtual database operation resultset can be operated on as a table via JDBC or another virtual database operation. Such a SQL view is static in two ways: first, if the virtual database operation’s schema changes, you must regenerate the SQL view accordingly; second, the removal of a virtual database operation’s SQL view can only be done as two separate steps—removing the old SQL view, followed by generating a new one. Here’s how you manage a virtual database operation’s SQL view:

- Step 1** Navigate to the View Virtual Database Operations screen:

Home > Data integration > Manage virtual database operations

View Virtual Database Operations							
Grid domain: <input type="text" value="CherylDomain"/>							
You are viewing the virtual database operations in the current domain.							
Name	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyVirtualDBOperation	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
<input type="button" value="Create Operation"/>		<input type="button" value="Done"/>					

- Step 2** Click the **Metadata** link beside the virtual database operation whose metadata you want to manage. The Manage Metadata screen appears. If there's not yet a SQL view representing this virtual database operation (but the operation's schema has been generated), the screen looks like this:

Manage Metadata	
Manage the metadata for virtual database operation CherylDomain.VirtualDb.MyVirtualDBOperation : generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.	
Schema: Refresh View	
Dependencies: Browse	
<input type="button" value="Done"/>	

SQL View	
Logical name: <input type="text"/>	Generate Remove

Enter a name for the SQL view to be generated, and click **Generate**. **Note:** Do not include a space in the name.

- Step 3** If there is a SQL view representing this database operation, the screen looks like this:

Manage Metadata	
Manage the metadata for virtual database operation CherylDomain.VirtualDb.MyVirtualDBOperation : generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.	
Schema: Refresh View	
Dependencies: Browse	
<input type="button" value="Done"/>	

SQL View	
Logical name: MyVirtualDBOperationTable	Generate Remove

To remove the SQL view, click the **Remove** link.

Executing virtual database operations

You can display the results of a virtual database operation to make sure that the operation is set up correctly and that the connections are healthy.

Follow these steps to execute a virtual database operation:

- Step 1** Navigate to the View Virtual Database Operations screen:

Home > Data integration > Manage virtual database operations

View Virtual Database Operations

Grid domain: CherylDomain ▼

You are viewing the virtual database operations in the current domain.

Name	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyVirtualDBOperation	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

Create Operation
Done

- Step 2** Click the **Execute** link beside the name of the database operation you want to execute. The Execute Database Operation page appears.

Execute Virtual Database Operation

Specify the values of the input parameters for virtual database operation **CherylDomain.VirtualDb.MyVirtualDBOperation**, and then click **Submit** to execute it.

SQL statement: select * from emp where ename = ?

Parameter 1: VARCHAR

Generate schema and suppress output:

Submit
Done

- Step 3** Optional. If your query requires parameters, enter the parameters.
- Step 4** Optional. Select the **Generate schema and suppress output** option to generate schema for the virtual database operation. The schema describes the structure of the output of a virtual database operation.
- Step 5** Click **Submit**. A confirmation screen appears. If you chose to generate schema in the previous step, you can click **View Schema** to view the virtual database operation's schema. If you chose not to generate schema, the results of the query appear in XML format.

Calling a virtual database operation

You can use ODBC, JDBC, or SOAP procedure calls to call a virtual database operation in the data catalog. For more information, see the *Sybase Avaki EII API Guide*.

Removing virtual database operations

Follow these steps to remove a virtual database operation:

- Step 1** Navigate to the View Virtual Database Operations screen:

Home > Data integration > Manage virtual database operations

View Virtual Database Operations

Grid domain: ▼

You are viewing the virtual database operations in the current domain.

Name	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyVirtualDBOperation	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

- Step 2** Click the **Remove** link beside the virtual database operation you want to remove. The system displays a confirmation screen.
- Step 3** Click **OK** to confirm the operation. The system redisplay the View Virtual Database Operations screen without the operation that you removed.

Managing virtual database services

The following sections describe how to browse the schemas in a virtual database and configure the virtual database service's access permissions.

Browsing virtual database schemas

When you're deciding which users and groups can create virtual database operations, it may be helpful to examine the schemas and tables in the virtual database service, as follows:

Step 1 Navigate to the Manage Virtual Database Service screen:

Home > Data integration > Manage virtual database service

Browse Virtual Schema

Allow/Disallow Users

The following users are allowed to create virtual database operations:

Name	Type	Domain	Auth Service	Auth Service Type	Remove
DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Remove
Administrators	Group	CherylDomain	DefaultAuthService	Grid	Remove

Activities

[Attributes](#) [Security](#)

Step 2 Database schema: Click the **View Schema** button to use the pop-up Database Table Browser to examine the tables defined in the Avaki virtual database. In JDBC terms, each Avaki domain is a “catalog,” which contains a variable number of schemas, which in turn may contain table definitions. There are four kinds of schemas in each domain:

- DATASERVICE: Contains all SQL views generated from data services.
- <Database-Connector-Name>: Contains all SQL views that are provisioned from the database to which the given database connector connects or are generated from database operations on this database connector.

- VIRTUALDB: Contains all SQL views generated from virtual database operations.
- <Metadata-Model-Name>: Contains all the mapped tables deployed from the given metadata model. Note that if a catalog other than the Avaki domain name was specified on deployment of a metadata model, that metadata model/schema and its tables will belong to the specified catalog, not to your domain's catalog. For more on metadata models, see *Data Integration with Sybase Avaki Studio*.

For more information about using the Database Table Browser, see [“Viewing database schemas” on page 10](#).

Configuring virtual database access permissions

The following sections describe how to configure access permissions for the Avaki virtual database service:

- [“Allowing users to create virtual database operations,”](#) below
- [“Allowing groups to create virtual database operations” on page 67](#)
- [“Preventing users from creating virtual database operations” on page 69](#)
- [“Preventing groups from creating virtual database operations” on page 70](#)

Allowing users to create virtual database operations

To allow a user to create virtual database operations in the Avaki virtual database, do the following:

- Step 1** Navigate to the Manage Virtual Database Service screen:

Home > Data integration > Manage virtual database service

Browse Virtual Schema

Allow/Disallow Users

The following users are allowed to create virtual database operations:

Name	Type	Domain	Auth Service	Auth Service Type	Remove
DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Remove
Administrators	Group	CherylDomain	DefaultAuthService	Grid	Remove

Activities

[Attributes](#) [Security](#)

- Step 2** To add a user to the list of users who can create virtual database operations in the Avaki virtual database, click the **Add User** button. The Add Virtual Database User screen appears.

Add Virtual Database User

All users
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

In the Add column, place a check mark next to the names of the user(s) you want to allow to create virtual database operations:

Add	Username	Authentication service	Authentication service type
<input type="checkbox"/>	Administrator	DefaultAuthService	Grid
<input type="checkbox"/>	MessagingUser	DefaultAuthService	Grid
<input type="checkbox"/>	wilma	DefaultAuthService	Grid

- Step 3** Click boxes in the Add column to select the users you want to add.
- Step 4** Click **Submit** to save your changes. The system displays a list of the users you have added.

Allowing groups to create virtual database operations

To allow a group to create virtual database operations in the Avaki virtual database, do the following:

- Step 1** Navigate to the Manage Virtual Database Service screen:

Home > Data integration > Manage virtual database service

Browse Virtual Schema

Allow/Disallow Users

The following users are allowed to create virtual database operations:

Name	Type	Domain	Auth Service	Auth Service Type	Remove
DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Remove
Administrators	Group	CherylDomain	DefaultAuthService	Grid	Remove

Activities

[Attributes](#) [Security](#)

- Step 2** To add a group to the list of groups who can create virtual database operations in the Avaki virtual database, click the **Add Group** button. The Add Virtual Database Group screen appears.

Add Virtual Database Group

All groups
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

In the Add column, place a check mark next to the names of the group(s) you want to allow to create virtual database operations:

Add	Name	Authentication service	Authentication service type
<input type="checkbox"/>	Administrators	DefaultAuthService	Grid
<input type="checkbox"/>	DataProviders	DefaultAuthService	Grid
<input type="checkbox"/>	DatabaseAdministrators	DefaultAuthService	Grid
<input type="checkbox"/>	DomainUsers	DefaultAuthService	Grid
<input type="checkbox"/>	MessagingUsers	DefaultAuthService	Grid
<input type="checkbox"/>	UserAdministrators	DefaultAuthService	Grid

- Step 3** Click boxes in the Add column to select the groups you want to add.
- Step 4** Click **Submit** to save your changes. The system displays a list of the groups you have added.

Preventing users from creating virtual database operations

To prevent a user from creating virtual database operations in the Avaki virtual database, do the following:

- Step 1** Navigate to the Manage Virtual Database Service screen:

Home > Data integration > Manage virtual database service

Browse Virtual Schema

Allow/Disallow Users

The following users are allowed to create virtual database operations:

Name	Type	Domain	Auth Service	Auth Service Type	Remove
DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Remove
Administrators	Group	CherylDomain	DefaultAuthService	Grid	Remove

Activities

[Attributes](#) [Security](#)

- Step 2** To remove a user from the list of users who can create virtual database operations in the Avaki virtual database, click **Remove**. The system removes the user from the list.

Preventing groups from creating virtual database operations

To prevent a group from creating virtual database operations in the Avaki virtual database, do the following:

- Step 1** Navigate to the Manage Virtual Database Service screen:

Home > Data integration > Manage virtual database service

Browse Virtual Schema

Allow/Disallow Users

The following users are allowed to create virtual database operations:

Name	Type	Domain	Auth Service	Auth Service Type	Remove
DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Remove
Administrators	Group	CherylDomain	DefaultAuthService	Grid	Remove

Activities

[Attributes](#) [Security](#)

- Step 2** To remove a group from the list of groups that can create virtual database operations in the Avaki virtual database, click **Remove**. The system removes the group from the list.

Configuring virtual database service attributes

The Avaki virtual database service has attributes that store information such as the time at which the virtual database service was created and the name of the user who owns the service.

To view the attributes for the virtual database service, do the following:

- Step 1** Navigate to the Manage Virtual Database Service screen:

Home > Data integration > Manage virtual database service

Browse Virtual Schema

Allow/Disallow Users

The following users are allowed to create virtual database operations:

Name	Type	Domain	Auth Service	Auth Service Type	Remove
DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Remove
Administrators	Group	CherylDomain	DefaultAuthService	Grid	Remove

Activities

[Attributes](#) [Security](#)

- Step 2** Click the **Attributes** link. The View Attributes page appears, showing the attributes for the Avaki virtual database service.

System Attributes		
Name	Type	Value
system/QualifiedName	string	CherylDomain.VirtualDb
system/GridServerPath	string	/System/Domains/CherylDomain/Servers/magadieu.avaki.local
system/VirtualDatabaseServicesPath	string	/System/Domains/CherylDomain/Servers/magadieu.avaki.local/Services/VirtualDb
system/ObjectHostName	string	magadieu.avaki.local:1299
system/ObjectHostIndiPort	string	1299
system/References	integer	1
system/ObjectId	string	loid:/1/1005428d9ad/1005428dab2/1005428db7c
system/ObjectType	string	Virtual Database Service
system/ChangeTime	timestamp	2004-12-02 22:00:06.65
system/ModificationTime	timestamp	2004-11-19 23:07:35.889
system/Owner	string	/System/Domains/CherylDomain/Services/AuthServices/Grnd/DefaultAuthService/Groups/Administrators
system/Description	string	Avaki Virtual Database

User-defined Attributes				
A user-defined attribute may be one of the following types:				
Type	Description	Format		
String	Text	Any characters		
Integer	Any whole quantity	Any integer		
Float	A numeric value that can be fractional or very large	Any numeric value		
Date	Year, month, and date	yyyy-mm-dd		
Time	Hour, minute and second that an event occurs	hh:mm:ss		
Timestamp	A precise time and date	yyyy-mm-dd hh:mm:ss.ffffff		
To modify an attribute's value, enter the new value and make sure the Update checkbox is selected. To delete an existing attribute, check the checkbox in its Delete column.				
Name	Type	Value	Update	Delete
dbconn/UselidentityMapping	string	<input type="text" value="false"/>	<input type="checkbox"/>	<input type="checkbox"/>
forcedDelete	string	<input type="text" value="true"/>	<input type="checkbox"/>	<input type="checkbox"/>

For details about creating, modifying, and deleting attributes, see the *Sybase Avaki EII Administration Guide*.

Managing virtual database service ACLs

An access control list (ACL) determines which grid users are allowed to read and manipulate the Avaki virtual database service.

To view the ACL for the virtual database service, do the following:

- Step 1** Navigate to the Manage Virtual Database Service screen:

Home > Data integration > Manage virtual database service

Browse Virtual Schema

Allow/Disallow Users

The following users are allowed to create virtual database operations:

Name	Type	Domain	Auth Service	Auth Service Type	Remove
DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Remove
Administrators	Group	CherylDomain	DefaultAuthService	Grid	Remove

Activities

[Attributes](#) [Security](#)

- Step 2** Click the **Security** link. The View Security Information appears, showing the users and groups that have been added to the ACL for the virtual database service.

View Security Information

You are viewing security information for the following object:
/System/Domains/CherylDomain/Services/VirtualDatabaseServices/Services/magadiou.avaki.local

The current owner of the object is the Group **Administrators** in domain **CherylDomain** and the Grid authentication service **DefaultAuthService**.

The following users and groups have been added to the Access Control List (ACL) for this object. To modify a user or group's permissions, place a check mark next to the user or group and click **Edit All Checked**. To add a user who is in the current grid domain, click **Add User to ACL**. To add a group that is in the current domain, click **Add Group to ACL**. To add a user or group that is in a connected domain, click **Add Via Interconnect ID**.

Select	Name	Type	Domain	Auth Service	Auth Service Type	Read	Write	Execute	Delete
<input type="checkbox"/>	DomainUsers	Group	CherylDomain	DefaultAuthService	Grid	Allow	Unset	Unset	Unset
<input type="checkbox"/>	Administrators	Group	CherylDomain	DefaultAuthService	Grid	Unset	Unset	Unset	Unset
<input type="checkbox"/>	wilma	User	CherylDomain	DefaultAuthService	Grid	Unset	Unset	Allow	Unset

Manage Access Control Lists

[Edit All Checked](#) Configure permissions for each user or group in the ACL for an object

[Add User to ACL](#) Add a user to the ACL for an object

[Add Group to ACL](#) Add a group to the ACL for an object

[Add Via Interconnect ID](#) Upload an interconnect ID to add a user or group to the ACL for an object

For details about viewing and modifying ACLs, see the *Sybase Avaki EII Administration Guide*.

Data services overview

This section explains what data services can do for you and how they're constructed. In this section:

- [“About data services,”](#) below
- [“Understanding data service components”](#) on page 76
- [“Avaki Studio and data services”](#) on page 78
- [“Data services and distributed transactions”](#) on page 78

About data services

A data service provides data to applications or users when they need it. Data services do the following:

- Data services can be used to integrate data from a variety of sources—flat files (especially CSV and XML files), databases, HTTP sources generally and web services in particular. The complexity of the integration is entirely hidden from applications using the data.
- Data services allow data to be published in arbitrary formats, with detailed control over access.
- Data services allow several database operations to be grouped together and executed as a single distributed transaction. (See [“Data services and distributed transactions”](#) on page 78.)
- Data services extend the benefits of Avaki caching, scheduling, and provisioning to web services data: they allow the data to be made available to applications in a timely manner while protecting the web services or the wide-area network from excessive load.
- As with Avaki database operations, the regular use of Avaki mechanisms to publish data makes the Avaki data catalog increasingly useful to the organization for finding data that has been made available and for understanding how that data is used.

One way to make data available is to publish data services in the data grid. A data service can be set up in any way that makes sense for the consuming application. For a dashboard application, you might set up one data service that fetches a single value for a performance metric. For another application, the data service might fetch an entire table, and in a third case the data service might supply data that's been combined from several data sources through an integration tool.

In addition to relational data, data services can be used with data that is available in XML files or in flat files. If data is coming into the organization from a partner or vendor, you can integrate it with your own data from relational and non-relational sources.

Data services can transform, integrate, and aggregate data in a number of different ways. To transform data, you can use Avaki Studio to build a data service, or you can write your own data service plug-in using XSLT, JavaScript, or Java code.

A data service can be used to create cross-functional applications such as:

- Corporate performance management dashboards, where data must be consolidated from multiple functional groups and divisions;
- Customer service applications, which consolidate customer data from multiple systems to create a single view;
- Reporting applications that aggregate data from multiple sources to provide the right rollups for decision-makers.

You can go to one catalog to retrieve data to pull into these applications—instruct data owners to publish catalog entries that you can use. Data owners keep control over who has access to data, and developers need to use only one set of interfaces. Data services support standard interfaces such as ODBC, JDBC, standard file I/O, and SOAP, so application developers don't have to worry about individual drivers or connections.

The data service feature does not depend on the existence of a Web services architecture. However, if your organization is standardizing on a Web services architecture, the data service layer can provide its data using SOAP and WSDL.

The Avaki data catalog (comprising data services, database operations, and provisioned SQL views) can help an organization bring data assets under control and get more use out of work that has been done in data management and data integration. The catalog makes visible what data is available, who uses it for what, and where it can be found.

Understanding data service components

A data service is a named data catalog entry that contains the name of a data service plug-in and an external interface definition of the data service's input and output.

A data service plug-in is the core of an Avaki data service. A plug-in is a logic module that can be written in Java, JavaScript, or XSLT. It contains the logic through which the input data is transformed, integrated, or aggregated. A plug-in defines the data service's requirements for the following:

- **Input sources:** An input stream can consist of XML, rowset, or raw binary data. The source can be a grid file, an HTTP source, a web service, or the result of a database operation or another data service.

Note It's important to understand your sources and how they are used. Avaki tracks the ways in which Avaki operations and objects depend on one another, and makes this information visible to you in a dependencies display; this information can be a useful aid to understanding. For information about checking data service dependencies, see [“Viewing data service dependencies” on page 97](#).

- **Runtime parameters:** Runtime parameters include parameters required by the data service plug-in, parameters required by input sources, and parameters required by the data service.
- **Output stream:** The plug-in's output stream may consist of a byte stream (provided to Java applications as an input stream) or a rowset or XML. A rowset is an Avaki internal format for relational results. Data services and other Avaki objects accept rowsets as input.

For more on data service components, see [Chapter 4, “Setting up data service plug-ins”](#).

To create a data service in the Avaki web UI, you specify which data service plug-in to use. Any existing plug-ins can be made available in the data catalog as files so that they are readily available to architects or developers creating data services.

Once you have specified which plug-in the new data service will use, Avaki knows which inputs the plug-in requires and can help you complete the definition of the data service by specifying the sources of those inputs.

Data services produce their data dynamically when executed. Data resulting from the execution of data services can be cached using Avaki's normal caching mechanism. The results are passed to the calling application, but they are not stored anywhere in the data grid unless caching is in effect.

You can cache the results of a data service to reduce the load on a back-end data source or cut down on network congestion and speed up application performance. For details about caching data services, see [Chapter 3, “Managing cache services”](#).

Data services are run by execution services. There is an execution service on every grid server, and you can configure a pool of execution services on a grid server. When a pool is in place, a data service can be run by any execution service in its grid server's pool. For more about configuring execution services, see the *Sybase Avaki EII Administration Guide*.

Avaki Studio and data services

Avaki Studio is a graphical tool whose primary purpose is the creation of data services. In Avaki Studio, you create *view models*, which (in terms of the preceding discussion) combine a data service plug-in with the specification of data service inputs and parameters; you then deploy those view models in Avaki as data services. Avaki Studio also lets you manage attributes of and access control for data services (whether or not those data services originated in Studio). (Scheduling data services, however, can be performed at present only in Avaki's web and command-line interfaces.)

Because of its graphical nature and rich user interface, Studio is the preferred way to create many data services. However, data services created in Studio are fundamentally relational in nature; while they accept nonrelational data, the first step in doing so in Studio is to set up a transformation on that data that yields a relational result. If you want your data service to work or produce results outside the relational paradigm, you'll need either to use one of the built-in plug-ins or to build one of your own. (You might use the built-in no-operation plug-in to provision a web service's data as XML, or the XSLT plug-in to use XSLT to process one or more XML inputs. For another format—a data service that does image processing, perhaps—you'd write your own plug-in.)

For more about Avaki Studio, see *Data Integration with Sybase Avaki Studio*.

Data services and distributed transactions

A *distributed transaction* is a set of related operations—typically SQL operations such as SELECT, INSERT, UPDATE, DELETE, and CALL—that

- involve one or more databases, and
- might lead to unwanted results (such as leaving participating databases in an inconsistent state or producing inconsistent reads) if some of the operations complete and others do not, and therefore
- must all be executed at once, as a single transaction.

For example, consider an application that posts credits and debits to customer accounts in one database and records account balances in another. Clearly, the balances can't be allowed to get out of sync with the credit and debit records. In a distributed transaction, all the operations must be completed—*committed*—or else all the operations are cancelled—*rolled back*. This ensures that all the databases involved are always left in a consistent state.

In Avaki, the individual operations that make up a distributed transaction are performed by database operations that rely on specially configured database connectors, and all the database operations are executed by a specially configured data service.

Requirements for distributed transactions

Avaki distributed transactions are subject to the following requirements and restrictions:

- Distributed transactions have been tested and found to work on these DBMSes:
 - Sybase ASE
 - Oracle 10g
 - MySQL

For a list of tested DBMS versions, see [page 7](#).

- All database operations participating in distributed transactions must use database connectors that are configured with XA drivers ([page 7](#)). (Such database operations will not use the XA driver when executed outside a distributed transaction data service.)
- To execute a distributed transaction, you must write a data service plug-in that uses Avaki's transaction API to run the participating database operations (“[Writing the Java code](#)” on [page 185](#)).
- The grid server where the data service runs must be able to establish network connections with the participating databases.

Two-phase commit protocol

A distributed transaction is accomplished by a data service that uses the *two-phase commit protocol* to execute a group of database operations as a single transaction. The two-phase commit protocol causes the operations in the transaction to be saved to temporary storage in the various databases (phase 1), then polls all the databases involved to make sure they're ready to commit the operations to permanent storage. Only when all the databases involved have confirmed that they're ready to commit is the command issued to complete the distributed transaction by saving the work to permanent storage (phase 2). If any database fails to confirm that it's ready, all the databases roll back their parts of the transaction, nothing is saved to permanent storage, and the data service returns an error.

This differs from the typical behavior of Avaki database operations. A database operation that has not been configured as part of a data service designed to perform distrib-

uted transactions is set up to *auto-commit* when its underlying SQL statement is executed. That means that when multiple database operations execute as part of a single data service, each database operation runs its own transaction. As a result, multiple database operations reading from the same database will not necessarily see the same view of the data. If one database operation fails, there's no way to roll back the others, and the underlying data may be left in an inconsistent state. The distributed transaction feature has been implemented to solve these kinds of problems.

The two-phase commit protocol is not infallible; for example, if one database involved in a distributed transaction crashes immediately after confirming that it's ready to commit, it will not receive the final command to commit and thus will not save its part of the transaction. But the other databases will save their parts of the transaction, and the database that crashed will be out of sync with the others. Similarly, if a crash or network failure affects the hosting grid server while a data service is running a distributed transaction, the transaction may be interrupted in a way that leaves the participating databases in an inconsistent state.

Setting up a distributed transaction

To create a distributed transaction data service, do the following:

1. Configure XA drivers on the database connectors that will be used by the database operations in the distributed transaction ([“Creating database connectors” on page 3](#)).
2. Set up the database operations that will participate in the distributed transaction ([“Creating database operations” on page 22](#)). **Note:** For purposes of the distributed transaction, any caching configured for participating database operations will be bypassed—the data service will execute all the database operations to obtain fresh results.
3. Write and deploy a Java data service plug-in that uses Avaki's transaction API to execute the database operations ([Chapter 4, “Setting up data service plug-ins”](#)).
4. Set up a data service that uses your data service plug-in ([“Creating data services” on page 81](#)).

Creating data services

The following sections describe how to create a data service in the Avaki web UI. In many cases it's easier to create data services in Avaki Studio; see *Data Integration with Sybase Avaki Studio*.

- [“Getting started,”](#) below
- [“Configuring data service plug-ins”](#) on page 82
- [“Configuring data service input parameters”](#) on page 84
- [“Configuring data service output streams”](#) on page 86
- [“Configuring data service input streams”](#) on page 87

Note Unless you are planning to use the no-operation plug-in to provision some data into Avaki without doing any processing on it, you'll need to create a data service plug-in (or at least the XSLT input to the XSLT plug-in). For instructions, see [“Setting up data service plug-ins”](#) on page 175.

Getting started

Whatever plug-in you plan to use, you'll always start as follows:

Step 1 Log in as a member of the DataProviders group.

Step 2 Navigate to the Create Data Service screen:

Home > Data integration > Create data service

Create Data Service

Choose the type of data service plug-in you wish to use and then click **Next**.

Use an XSL style sheet to transform data.

Use JavaScript to transform data.

Use a custom data service plug-in from a JAR file in the grid.

Use a no-operation plugin that does not transform data.

Configuring data service plug-ins

Before you configure a data service plug-in in the web UI, you must create and deploy the plug-in. See [“Setting up data service plug-ins” on page 175](#) for instructions on creating data service plug-ins.

To configure a data service plug-in, do the following:

- Step 1** Choose the type of data service plug-in you wish to use: an XSL style sheet, JavaScript, a custom Java data service plug-in from a JAR file in the data catalog, or the no-operation plug-in that does not transform data.

Note If you create a data service that uses a JAR file plug-in, the startup performance for the data service will be faster if the JAR file is pinned (marked) for scheduled caching. For information about how to pin files, see [Chapter 3, “Managing cache services”](#).

Create Data Service

Choose the type of data service plug-in you wish to use and then click **Next**.

Use an XSL style sheet to transform data.

Use JavaScript to transform data.

Use a custom data service plug-in from a JAR file in the grid.

Use a no-operation plugin that does not transform data.

- If you are using an XSL plug-in, go to [Step 2](#).
- If you are using a JavaScript plug-in, skip to [Step 4](#).
- If you are using a custom plug-in from a JAR file, skip to [Step 6](#).
- If you are using a no-operation plug-in, skip to [“Configuring data service input parameters” on page 84](#).

Step 2 If you selected the XSL Style Sheet plug-in, fill in the Path to XSL and XSLT Engine fields:

- Path to XSL File: Enter or browse for the grid path to the style sheet you want to use.
- XSLT Engine: From the pull-down list, select an XSL engine to use for the data service. Avaki provides two engines to choose from: Saxon or Xalan.

The screenshot shows a dialog box titled "Create Data Service". The instruction text reads: "Specify the style sheet to use, as well as the XSLT engine, and click **Next**." Below the text, there is a text input field labeled "Path to XSL File:" followed by a "Browse..." button. Below that is a dropdown menu labeled "XSLT Engine:" with "Saxon" selected. At the bottom, there are two buttons: "Next" and "Cancel".

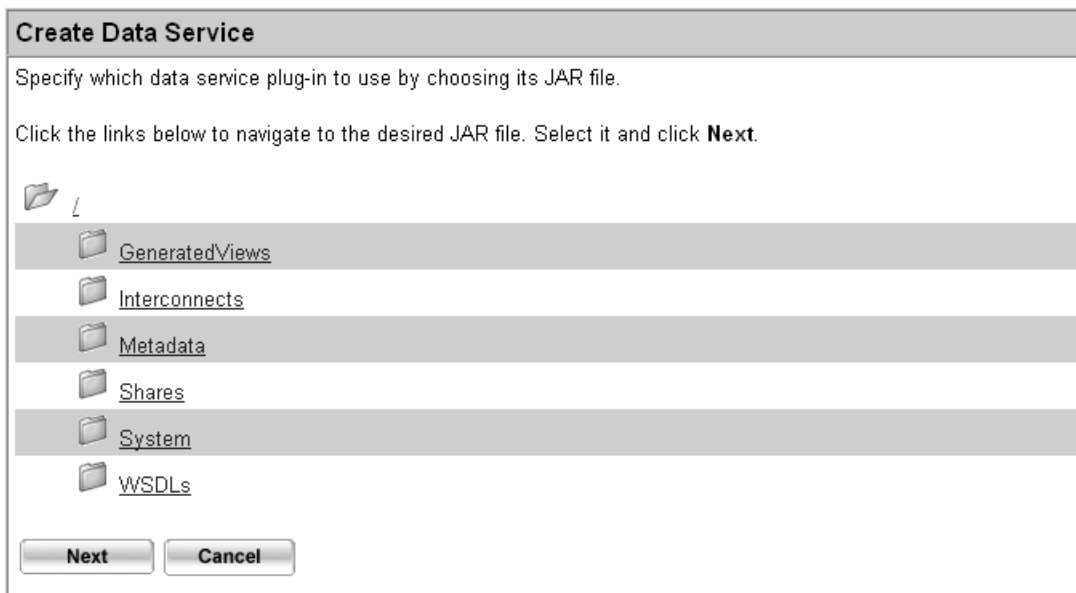
Step 3 Click **Next** and skip to [Step 6](#).

Step 4 If you selected the JavaScript plug-in, enter or browse for the grid path to the JavaScript file you want to use.

The screenshot shows a dialog box titled "Create Data Service". The instruction text reads: "Specify the JavaScript file to use and click **Next**." Below the text, there is a text input field labeled "Path to JavaScript file:" followed by a "Browse..." button. At the bottom, there are two buttons: "Next" and "Cancel".

Step 5 Click **Next**.

- Step 6** In the data catalog, navigate to the JAR file share that contains your data service plug-in. Select the JAR file.



- Step 7** Click **Next**.

Configuring data service input parameters

After you configure a data service plug-in, you can configure one or more data service input parameters, as follows:

- Step 1** After you configure a data service plug-in, the screen that appears shows the plug-in that you chose and provides a box where you can specify input streams for the data service. If you want to view the full path of the data service plug-in, place your mouse over the name of the plug-in.

If the data service plug-in is an XSL style sheet, you must specify a primary input stream, and you can specify one or more secondary input streams. If you are using a

JavaScript plug-in or a Java plug-in from a JAR file, you can specify multiple input streams, and there is no hierarchy among the streams.

Create Data Service

No Parameters

Input Stream
Name: Primary Input
Stream Type: XML
From:
Target: Not set
[Edit Stream](#)

Input Stream
Name: Secondary Inputs
Stream Type: XML
Instances: 0 [Add](#)

Plug-in
Name: to-html.xsl
Description: XSLT transform using Saxon
Coherence Window: 5 minutes
[Change Plug-in](#)

Input Parameter
Name: HeadCellColour
Type: VARCHAR
Description: XSLT Stylesheet parameter
Value: Not set
[Edit Parameter](#)

Input Parameter
Name: Title
Type: VARCHAR
Description: XSLT Stylesheet parameter
Value: Not set
[Edit Parameter](#)

Input Parameter
Name: CellColour
Type: VARCHAR
Description: XSLT Stylesheet parameter
Value: Not set
[Edit Parameter](#)

Output Stream
Name: Result: XML

[Next](#) [Cancel](#) [Export Descriptor](#)

- Step 2** To specify an input parameter, click on the Edit Parameter link in the box for the parameter.
- Step 3** In the screen that appears, select one of the following options to configure the value for the input parameter:
- **Use this value:** Specify a static value to use whenever the data service is executed.
 - **Use an existing parameter to the data service:** This option is available only if there is already an existing dynamic parameter in the data service. Any existing parameters appear on the top left of the Create Data Service screen.
 - **Create a new parameter to the data service:** To create a new dynamic parameter, specify a name and description for a value that the user will specify when the data service is executed.

- **Do not specify a value for this parameter:** Select this option if the user will not specify a value when the data service is executed.

Update Data Service

Specify how the value for the plug-in parameter should be obtained.

Parameter Name: CellColour
Type: VARCHAR

Value: Use this value:

Use an existing parameter to the Data Service.
Param:

Create a new VARCHAR parameter to the data service.
Name:
Description:

Do not specify a value for this parameter.

- Step 4** Click **Submit**. If you specified a static value for the parameter, the parameter appears in the list of existing parameters at the top left of the Create Data Service screen.
- Step 5** Optional. To delete an existing input parameter, click the Remove Parameter link in the box for the parameter.

Parameter

Name: EmpName
Type: VARCHAR
Description: Employee name

[Remove Parameter](#)

Configuring data service output streams

The data service output stream is where the data service writes the results of the data operation that the plug-in performs. Select the desired output stream from the pull-down list in the Output Stream box on the Create Data Service screen.

Output Stream

Name:

The output stream can be one of the following:

- XML: The output stream contains well-formed XML.
- ByteStream: The output stream contains bytes.

- **ResultSet:** The output stream contains database results rowsets.

Configuring data service input streams

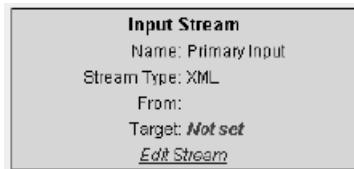
A data service that is based on an XML style sheet plug-in requires a primary input stream, and you may also specify secondary input streams. A data service that is based on a Java plug-in can have any number of input streams, and there is no hierarchy among the streams.

For the data service's input stream, specify one of the following:

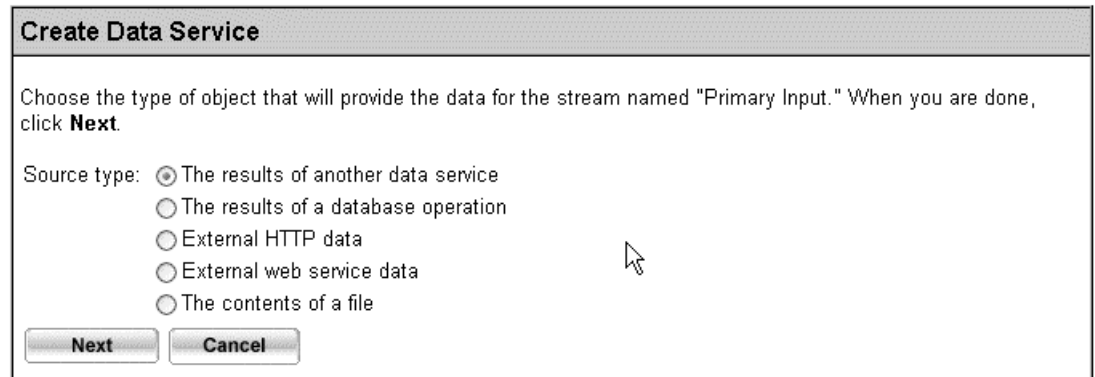
- the results of another data service;
- the results of a database operation;
- the contents of a file.

To specify the input stream, do the following:

- Step 1** On the Create Data Service screen, click the Edit Stream link for the box whose input stream you want to specify. For example, if the data service uses an XML style sheet plug-in, click the Edit Stream in the box labeled "Primary Input."



- Step 2** In the screen that appears, select the type of object that will provide the data for the input stream, then click **Next**.



- Step 3** Select the data service, database operation, or file that provides the data for the input stream, then click **Continue**. The screen that appears shows the parameters that the

input stream requires. In the following example, the database operation MyDBOperation requires one SQL input stream parameter, of type VARCHAR.

Input Stream Parameter
Name: SQL parameter 1
Type: VARCHAR
Description: An ordinal SQL parameter
Value: <i>Not set</i>
Edit Parameter

Step 4 For each input stream parameter, you can specify one of the following:

- a single SQL parameter to pass to the input stream; or
- a grid file that contains several SQL parameters to pass in to the input stream (this option is available if the database operation supports batch processing; for more information about batch processing, see [“Managing information from databases” on page 1](#)).

If you are *not* specifying a single SQL parameter, skip to [Step 5](#).

To specify a single SQL parameter, click the Edit Parameter link in the first Input Stream Parameter box.

In the screen that appears, select one of the following options to configure the value for the input stream parameter:

- **Use this value:** Specify a static value to use whenever the data service is executed.
- **Use an existing parameter to the data service:** This option is available only if there is already an existing dynamic parameter in the data service. Any existing parameters appear on the top left of the Create Data Service screen.
- **Create a new parameter to the data service:** To create a new dynamic parameter, specify a name and description for a value that the user will specify when the data service is executed.

- **Do not specify a value for this parameter:** Select this option if the user will not specify a value when the data service is executed.

Create Data Service

Specify how the value for the stream parameter should be obtained.

Parameter Name: SQL parameter 1
 Type: VARCHAR

Value: Use this value:

Use an existing parameter to the Data Service.
(There are no data service parameters of type VARCHAR.)

Create a new VARCHAR parameter to the data service.
 Name:
 Description:

Do not specify a value for this parameter.

Click **Submit**. If you specified a static value for the parameter, the parameter appears in the list of existing parameters at the top left of the Create Data Service screen.

Step 5 Optional. To specify a grid file that contains several SQL parameters to pass in to the input stream, click the Edit Parameter link in the Input Stream Parameter box named “Input File.” In the screen that appears, select one of the following options to configure the value for the input stream parameters:

- **Use this value:** Specify a static value to use whenever the data service is executed.
- **Use an existing parameter to the data service:** This option is available only if there is already an existing dynamic parameter in the data service.
- **Create a new parameter to the data service:** To create a new dynamic parameter, specify a name and description for a value that the user will specify when the data service is executed.
- **Do not specify a value for this parameter:** Select this option if the user will not specify a value when the data service is executed.

After you specify the grid file, click **Submit**. If you specified a static value for any parameters, the parameters appear in the list of existing parameters at the top left of the Create Data Service screen.

Step 6 Optional. To configure an additional input stream, do the following:

- If the data service uses an XML style sheet plug-in, click the **Add** link in the Input Stream box named Secondary Inputs.

- If the data service uses a Java or JavaScript plug-in, click the **Add** link in the Input Stream box for the input stream you specified in [Step 4](#).
 - To specify the grid object that will provide the data for the input stream, click the **Edit Stream** link in the Input Stream box.
 - In the screen that appears, enter a name for the input stream.

Create Data Service

You are adding a stream instance to the input stream named "Secondary Inputs". Please enter a name for the new instance. The data service plug-in will use this name to access the stream. When you are done, click **Next** to add the instance.

Instance Name:

- Select the data service, database operation, or file that provides the data for the input stream, then click **Next**.

Create Data Service

Choose the type of object that will provide the data for the instance "Secondary Input 1" of the stream named "Secondary Inputs." When you are done, click **Next**.

Source type: The results of another data service
 The results of a database operation
 External HTTP data
 External web service data
 The contents of a file

- Step 7** Click **Next** to continue creating the data service, or click **Export Descriptor** if you would like to save a descriptor for an incomplete data service and finish creating the data service later. If you click **Export Descriptor**, on the screen that appears, specify a path where you would like to save a descriptor for this data service. For information about importing a saved descriptor, see [“Importing data service descriptors” on page 92](#).

Step 8 On the screen that appears, enter a name for the data service.

Create Data Service

Choose a name for the data service and the server where you would like to create it. You may also specify a value for the expiration of the data service's data and for the coherence window for the data service's plug-in. When you are done, click **Next** to create the data service. You may also click **Export Descriptor** to save the current state of this data service as a data service descriptor file and continue creating it at a later time.

Or click **Return to Overview** to return to editing the name, description, parameters, input stream(s), plug-in, or output type of this data service.

Data service name:

Description:

Grid server: ▼

Cached data expiration: No caching
 Never expires
 Expires after ▼

Plug-in coherence window: ▼

Execute data service as: The user calling the data service
 A specific user:
(e.g. `userName@authService.authServiceType.domain`)

Step 9 Optional. In the Description field, enter a description for the data service.

Step 10 Choose the grid server where you would like to create the data service. Specify a value for the expiration of the data service's results and the coherence window for the data service's plug-in.

- From the grid server pull-down menu, select a grid server for this data service.
- Cached data expiration: Select one of the following to indicate whether the data is cached and, if so, when the data expires from the cache:
 - No caching: The data is not cached.
 - Never expires: The data never expires from the cache.
 - Expires after *n* seconds: Specify the interval (in seconds, minutes, hours, or days) before the data expires from the cache.
- In the plug-in coherence window field, specify the duration (in seconds, minutes, hours, or days) during which the contents of the data service are assumed to be

fresh after the cache service has last inspected the back-end source object for updates. The default is 5 minutes.

- Select the user to execute the data service as: the user calling the service or a specific user. If you choose to execute the data service as a specific user, specify the qualified user name of a grid user who has permission to execute this database service. Use the following format:

```
<user-name>@<authservice>.<authservicetype>.<domain>
```

Note The Avaki web UI employs a special browsing feature when you're selecting the run-as user for a data service, database operation, or view. To make user browsing work properly, make sure the smooth scrolling option in your web browser is turned off. In Firefox or Internet Explorer, select Tools > (Internet) Options > Advanced and uncheck "Use smooth scrolling."

For example:

```
wilma@DefaultAuthService.Grid.Bedrock
```

Step 11 Click **Next** or **Export Descriptor** to finish creating the data service.

Importing data service descriptors

If you have exported a data service descriptor, do the following to resume working with it:

Step 1 Navigate to the Import Data Service Descriptor screen:

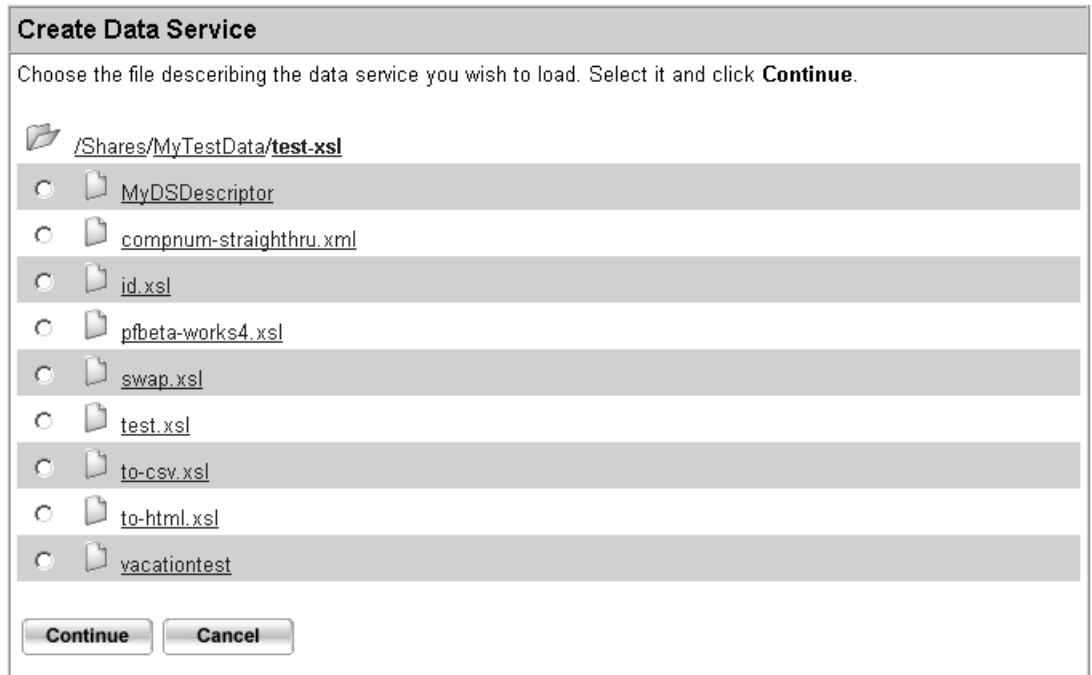
Home > Data integration > Import data service descriptor

Import Data Service Descriptor

Specify the path to the saved data service descriptor you wish to import. Then click **Import** to load it.

Path:

- Step 2** Click **Browse** and navigate to the data service descriptor you want to import, then select it.



- Step 3** Click **Continue**.

- Step 4** Click **Import** to import the data service descriptor and continue creating it.

Viewing a list of data services

To view a list of the data services in the current grid domain, navigate to the View Data Services screen:

Home > Data integration > Manage data services

View Data Services

Grid domain: ▼

You are viewing the data services in the current domain.

Name	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata
MyDataService	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata

The system displays a list of the data services in the current Avaki domain.

To view more information about a particular data service, click the **View/Edit** link beside the name of the data service whose details you want to view.

Modifying data services

Follow these steps to modify the settings for a data service:

- Step 1** Log in as a member of the Administrators or DataProviders group.
- Step 2** Navigate to the View Data Services screen:

Home > Data integration > Manage data services

View Data Services

Grid domain: ▼

You are viewing the data services in the current domain.

Name	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata
MyDataService	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata

- Step 3** Click the **View/Edit** link beside the name of the data service whose details you want to view. The system displays detailed information about the data service.

Update Data Service

No Parameters

Input Stream

Name: Primary Input
Stream Type: XML
From: File
Target: /Shares/MyTestShare2/test-xslt/test.xml
[Edit Stream](#)

Input Stream

Name: Secondary Inputs
Stream Type: XML[]
Instances: 0 [Add](#)

Plug-in

Name: to-html.xsl
Description: XSLT transform using Saxon
Coherence Window: 5 minutes
[Change Plug-in](#)

Input Parameter

Name: HeadCellColour
Type: VARCHAR
Description: XSLT Stylesheet parameter
Value: #D2D2D2
[Edit Parameter](#)

Input Parameter

Name: Title
Type: VARCHAR
Description: XSLT Stylesheet parameter
Value: Vacations
[Edit Parameter](#)

Input Parameter

Name: CellColour
Type: VARCHAR
Description: XSLT Stylesheet parameter
Value: #E2E2E2
[Edit Parameter](#)

Output Stream

Name:

- Step 4** Modify the data service settings as needed.
- Step 5** Click **Next** to continue, or click **Export Descriptor** to specify a path where you would like to save a descriptor for an incomplete data service and finish creating the data service later.

- Step 6** If you clicked **Next** in the previous step, the second Update Data Service screen appears.

Update Data Service

You may change the value for the expiration of the data service's data and/or the coherence window for the data service's plug-in. When you are done, click **Next** to update the data service.

Or click **Return to Overview** to return to editing the parameters, input stream(s), plug-in or output type of this data service.

Data service name: MyDataService

Description:

Cached data expiration: No caching
 Never expires
 Expires after seconds

Plug-in coherence window: 5 minutes

Execute data service as: The user calling the data service
 A specific user:
(e.g. userName@authService.authServiceType.domain)

Modify the settings as needed.

- Step 7** Click **Next** or **Export Descriptor** to save the revised data service.

Managing data service metadata

You can browse a data service's dependencies. If the data service's output is a result set, you can also generate or regenerate its schema (and view it if it has been generated) and expose the results of the data service as a SQL view.

Viewing data service dependencies

You can view a list that combines the data services, SQL views and view generators (if any) that depend on some data services and the data services and database operations that depend on it. To view these dependencies, do the following:

Step 1 Navigate to the View Data Services screen:

Home > Data integration > Manage data services

View Data Services							
Grid domain: <input type="text" value="CherylDomain"/> ▼							
You are viewing the data services in the current domain.							
Name	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata
MyDataService	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata
<input type="button" value="Create Data Service"/>		<input type="button" value="Done"/>					

Step 2 Click the **Metadata** link beside some data service. The Manage Metadata screen appears in one of the following two forms. If the data service's output is not a result set, its Manage Metadata screen is very limited:

Manage Metadata
Browse the dependencies of data service CherylDomain.MyWebDS.
Dependencies: Browse
<input type="button" value="Done"/>

If on the other hand the data service's output is a result set, its Manage Metadata screen contains more options (like those on the comparable screen for a database operation):

Manage Metadata

Manage the metadata for data service **CherylDomain.MyDataService**: generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.

Schema: [Generate](#) [View](#)

Dependencies: [Browse](#)

SQL View

You must generate this data service's schema before you can expose it as a SQL view.

Step 3 Either way, to view data service dependencies, click **Browse**.

Step 4 The system lists the data services, SQL views, and view generators that depend on the data service, as well as the data services and database operations it depends on.

Dependencies for CherylDomain.MyDataService		
Operation	Dependency	Dependency Type
Database Operation	CherylDomain.MyDBConnector.MyDBOperation	Input to CherylDomain.MyDataService
SQL View	CherylDomain.DATASERVICE.MyDataServiceTable	Input from CherylDomain.MyDataService

For details about creating view generators, see [Chapter 6, “Managing views” on page 217](#). For details about generating SQL views from data services, see [“Exposing data service results as a SQL view” on page 101](#).

Generating a data service's schema

To generate schema information for a data service that returns a result set, do the following:

Step 1 Navigate to the View Data Services screen:

Home > Data integration > Manage data services

View Data Services

Grid domain:

You are viewing the data services in the current domain.

Name	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata
MyDataService	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata

Step 2 Click the **Metadata** link beside the data service whose metadata you want to manage. The Manage Metadata screen appears.

Manage Metadata

Manage the metadata for data service **CherylDomain.MyDataService**: generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.

Schema: [Generate](#) [View](#)

Dependencies: [Browse](#)

Step 3 To generate schema information, click **Generate**.

Step 4 The Manage Metadata screen will change to indicate that the schema information has been generated:

Manage Metadata

Manage the metadata for data service **CherylDomain.MyDataService**: generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.

Schema: [Refresh](#) [View](#)

Dependencies: [Browse](#)

Step 5 Optional. To examine the data service's schema, click **View**.

CherylDomain.MyDataService									
Columns:									
Index	Name	Type	Precision	Scale	Display Size	Allows Nulls	Auto Increment	Case Sensitive	Read-Only
1	EMPNO	NUMERIC	4	4	22	false	false	false	false
2	ENAME	VARCHAR	10	10	10	true	false	true	false
3	JOB	VARCHAR	9	9	9	true	false	true	false
4	MGR	NUMERIC	4	4	22	true	false	false	false
5	HIREDATE	DATE	0	0	7	true	false	false	false
6	SAL	NUMERIC	7	7	22	true	false	false	false
7	COMM	NUMERIC	7	7	22	true	false	false	false
8	DEPTNO	NUMERIC	2	2	22	true	false	false	false

Done

Step 6 Optional. To regenerate schema information, click **Refresh**. You should regenerate the schema information if the structure of the data service's output changes. In that case, you should also propagate the change to any data services or SQL views that depend on the changed data service. For information about displaying data service dependencies, see [“Viewing data service dependencies,”](#) above.

Exposing data service results as a SQL view

You can expose data service results as a SQL view, so that the resultset can be operated on as a table via JDBC or a virtual database operation. Such a SQL view (which is sometimes called a *generated SQL view*) is static in two ways: first, if the data service's schema changes, you must regenerate the SQL view accordingly; second, the regeneration of a data service's SQL view can only be done as two separate steps—removing the old SQL view, followed by generating a new one.

Here's how to generate a SQL view from a data service:

Step 1 Navigate to the View Data Services screen:

Home > Data integration > Manage data services

View Data Services

Grid domain: ▼

You are viewing the data services in the current domain.

Name	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata
MyDataService	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata

Step 2 Click the **Metadata** link beside the data service whose metadata you want to manage. The Manage Metadata screen appears.

Manage Metadata

Manage the metadata for data service **CherylDomain.MyDataService**: generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.

Schema: [Refresh](#) [View](#)

Dependencies: [Browse](#)

SQL View

Logical name: [Generate](#) [Remove](#)

Enter a name for the SQL view to be generated, and click **Generate**. **Note:** Do not include spaces in the name.

Step 3 If there is a SQL view representing this data service, the screen looks like this:

Manage Metadata

Manage the metadata for data service **CherylDomain.MyDataService**: generate or regenerate its schema (and view it if it has been generated), expose it as an Avaki SQL view (virtual table), and browse its dependencies.

Schema: [Refresh](#) [View](#)

Dependencies: [Browse](#)

SQL View

Logical name: MyDataServiceTable [Generate](#) [Remove](#)

To remove the SQL view, click the **Remove** link.

Testing data services

Follow these steps to test whether a data service is operational:

Step 1 Navigate to the View Data Services screen:

Home > Data integration > Manage data services

View Data Services

Grid domain:

You are viewing the data services in the current domain.

Name	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata
MyDataService	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata

- Step 2** Click the **Test** link beside the data service you want to test. If the data service requires parameters, specify the values.

Test Data Service

Specify the values of the input parameters for this data service and then click **Submit** to execute the service.

Name	Type	Description	Value
GridFileTest	GridFile	DSTest	<input style="width: 100%;" type="text"/>

- Step 3** Click **Submit**. The system displays a message indicating whether the data service is operational.

Removing data services

Follow these steps to remove a data service:

- Step 1** Navigate to the View Data Services screen:

Home > Data integration > Manage data services

View Data Services

Grid domain:

You are viewing the data services in the current domain.

Name	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata
MyDataService	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata

- Step 2** Click the **Remove** link beside the data service you want to remove. The system displays a confirmation screen.
- Step 3** Click **OK** to confirm the operation. The system redisplay the View Data Services screen without the data service that you removed.

Managing cache services

Each Avaki grid server contains a *cache service* that can be used to store remote database results, application data, or files that are frequently accessed by users. Avaki uses caching to accomplish the following goals:

- insulate production data sources from random access;
- maintain good performance for users and applications;
- refresh data in a granular way based on business needs; and
- ensure maximum data availability.

Caching makes remote data access practical by limiting the number of times a data request requires immediate communication with the original data source. Avaki provides a variety of different caching options and features to meet diverse performance requirements. You can specify a different caching strategy for each data item, and the various caching options can be used separately or in combination to accomplish your goals.

- **Local caching** enables caching of frequently requested results near the data source to reduce load on the back-end data source.
- **Remote caching** caches data close to the users or applications that will use it. This cuts down on network congestion and dramatically speeds up application performance, because remote data calls are eliminated. Caches can be prepopulated and updated during off-hours when network load is low, and cache configurations can be established that ensure high availability when a network is congested or unavailable for some reason.

Cache update frequency can be specified for a given data item. Database administrators can schedule how often data services should be rerun and cached, to protect production databases from unexpected load. Cached data can also expire, after a set time period, forcing a refresh of the data on the next request.

Note A DGAS uses its own internal caches to store copies of the files and directories it serves to NFS and CIFS clients and other information. The DGAS caches may get their data directly from the source file systems or from the cache service associated with the DGAS. For information about configuring a DGAS's internal cache, see the *Sybase Avaki EII Administration Guide*.

The chapter covers the following topics:

- [“Configuring clients and Avaki servers to use cache services,”](#) below
- [“Configuring caching for files”](#) on page 107
- [“Configuring caching for database operations and data services”](#) on page 108
- [“Associating Avaki servers with caches”](#) on page 111
- [“Overriding cache service default settings”](#) on page 119
- [“Managing caches”](#) on page 120

Configuring clients and Avaki servers to use cache services

Avaki clients (such as command line clients) and applications that use the Avaki JDBC driver can be enabled to take advantage of cache services. To enable them to take advantage of remote caching, you must set the cache service that they will use. If you are using only local caching for database operations and data services, you do not need to perform any special configuration steps.

For information about configuring remote caching for JDBC programs, see the *Sybase Avaki EII API Guide*.

For information about configuring remote caching for Avaki command clients, see the *Sybase Avaki EII Administration Guide* or the *Sybase Avaki EII Command Reference*.

Avaki grid servers and data grid access servers can also be configured to use a particular cache service in a remote-caching configuration. For information about configuring remote caching for a grid server or DGAS, see the *Sybase Avaki EII Administration Guide*.

Configuring caching for files

There are two modes in which an Avaki cache service can cache the file data and metadata for shared files: on-demand caching and scheduled (pinned) caching. Both are discussed here.

On-demand caching

The cache coherence window determines how frequently a cache service will consult a file to see if it has changed. For files, this is determined by consulting the modification time (mtime) for the file or directory.

The cache service's coherence window controls cache coherence for the on-demand cached files in the current cache service. This value can be overridden on a per-file basis by coherence windows set on individual files (either as an attribute on the back-end file that dictates its coherence for all cache services, or during the tagging process that dictates its coherence for that file on that particular cache service). For information on setting coherence windows, see

- [“Viewing and modifying cache service configuration” on page 116](#) (per-cache)
- [“Overriding cache service default settings” on page 119](#) (per-file)

In order for a cache service to cache files on demand, you must specifically identify the file or files that should be cached on demand. Thus, a cache service will not automatically cache every single file that it may be asked for; it will cache only those that have been marked for on-demand caching. You can mark individual files or directory hierarchies of files for on-demand caching (see [“Caching files or directories on demand” on page 130](#)). When you mark a directory (and potentially its subdirectories) for on-demand caching, the cache service will track the directories and their contents, but not pull down file content. When a client to the cache service—a DGAS, command line interface, or SOAP client—asks for file content, the cache service pulls down and caches file content in blocks as needed.

Pinning files for scheduled caching

The alternative to on-demand caching is to pin files in a cache service for scheduled caching. As with on-demand caching, pinning involves marking individual files or directory hierarchies of files to be pinned. If you specify a schedule, the cache service pulls down file content during the pinning process and actively keeps the file content refreshed in the cache according to the schedule. If you do not specify a schedule, the cache service uses a schedule based on the applicable coherence window for that file. When the time specified for the coherence window elapses, the cache service consults the modification time for files and directories and syncs down any changes.

For instructions on pinning files for scheduled caching, see [“Caching files or directories on a schedule” on page 121](#).

Permissions and access control

Files and directories are pinned using the identity of the individual who marked them. Like data services and database operations, the cache service caches access control information and performs local access control checks. ACLs and other metadata is refreshed according to a cache coherence window interval.

Configuring caching for database operations and data services

There are two modes in which an Avaki cache service can cache the results-data and metadata for database operations and data services: on-demand caching and scheduled caching.

On-demand caching

On-demand caching is suitable for grid objects that are rarely accessed or that change at irregular intervals. When you tag a grid object for on-demand caching, the object is cached only if it is used—for example, results are cached when a database operation or a data service is executed, or a file is cached when a user or application reads it.

Cache coherence in Avaki is maintained via coherence window mechanisms. For database operations and data services, you can use the data expiration interval to control data freshness.

The data expiration interval is a settable property for data services and database operations. You can set the value of this property using either the Web user interface or the command line interface.

This property determines whether data is cached and, if so, when the data expires from the cache. You can specify that the data should never expire, or you can specify the interval before the data expires.

The default data expiration interval for a database operation or data service is no caching. You can change this data expiration interval when the database operation or data service is created or at any other time by using the view/edit option when viewing a list of database operations or data services. When the database operation or data service is executed, any results will be held in the cache for a period equal to the data expiration interval. Caching is keyed by database operation or data service name and parameter values, so any invocations of the database operation or data service with the same parameter values will be read from the cache during that time. Note that this means that multiple results for a particular database operation or data service may be in the cache at any given time.

When the results of a database operation or data service are being cached on demand, other metadata for the database operation or data service may be cached as well. This includes security information and attributes. If these values are changed while data is being cached, we recommend that you invalidate the database operation or data service in the cache service to force a reset of all cached data for that database operation or data service. Cached data can be invalidated on the View Cache Service UI page.

Scheduled caching

Scheduling is useful if you want a database operation, virtual database operation, or data service to run at a particular interval (minutes, hours, days, and so on). When a database operation or data service is scheduled, the results-data stays in the cache until the next scheduled execution takes place. During this time, the cache service assumes the content to be fresh and does not check with the source to see if the content has expired. The scheduling for a database operation or data service applies only to the cache service on which the schedule was created. The data expiration interval will be ignored (in the specific cache) for the database operation or data service when it has been scheduled.

For instructions on configuring scheduled caching:

- For database operations and virtual database operations, see [“Caching database operations on a schedule” on page 139](#)

- For data services, see [“Caching data services on a schedule”](#) on page 152

Remote/local caching interactions

The simplest caching behavior is when you are using only a local cache and you invoke a database operation or data service that has a data expiration interval higher than zero. The first time you invoke such a database operation or data service, a cache entry will be created. For the duration of the data expiration interval, data will be read from the cache.

With multiple caches, a user must be a little more careful about applying a strategy. The simplest scenario is to use on-demand caching only, by controlling the data expiration interval. In this case, the local cache and the remote cache will treat the data in the same manner. The data will expire in both caches at the same time and will always be fresh relative to the data expiration interval.

If you are implementing scheduling, you should understand which cache is taking the direct user requests before deciding on a strategy. The scheduling should be done in the cache that is taking the user requests—usually the remote cache. If both local and remote caches are taking user requests, scheduling in both caches may be useful. However, if you’re using a combination of scheduling and on-demand caching, use scheduling in the remote cache only. If you use scheduling in both caches in such a configuration, the remote cache may not work correctly because it can receive repeated stale data from scheduled database operations or data services in the local cache and will never actually cache the data.

Permissions and access control

By default, you need to be a member of the Administrators group in order to configure cache services.

When you schedule database operations or data services in a particular cache, the user identity that the cache uses to invoke the database operation or data service is the identity that was used to create the schedule. So if you are scheduling a database operation or data service, you must make sure that you have execute permission on it.

When a database operation or data service is accessed via a cache service, the cache service performs the access control check for the user who is invoking the service. In order to do this, the cache maintains a cached version of the object’s access control list (ACL). The cache service will refresh the ACL either on a schedule or according to the cache coherence window interval.

For information about configuring caching for database operations, see “[Managing database caches](#)” on page 139. For information about configuring caching for data services, see “[Managing data service caches](#)” on page 152.

Associating Avaki servers with caches

Grid servers and data grid access servers can be associated with Avaki cache services. When a server is associated with a cache service, the server uses the pinned directories or files and scheduled data services that are stored in the cache so they can be accessed quickly.

Associating grid servers with caches

To associate a grid server with a cache, you must have write permission on the grid server that you are associating with the cache service.

Follow these steps to associate a grid server with a cache:

- Step 1** Navigate to the View Grid Servers screen:

Home > Server management > View grid servers

View Grid Servers				
You are viewing the grid servers in the current grid domain.				
Server Name	View/Edit	Destroy Server	Attributes	Security
FRANCIUM.avaki.local	View/Edit	Destroy	Attributes	Security
cmagadieu.avaki.local	View/Edit	Destroy	Attributes	Security
<input type="button" value="Done"/>				

- Step 2** Click the **View/Edit** link beside the grid server that you want to associate with a cache. The View/Edit Grid Server screen appears.

View/Edit Grid Server

To associate the grid server with a cache service, select the cache service from the pull-down list below. Once a cache is set, the grid server uses it to store pinned files and scheduled database operations so they can be accessed quickly.

Server name: magadiou.avaki.local

Current cache service: None

New cache service:

- Step 3** From the “New cache service” pull-down menu, select the name of the cache service with which this grid server will be associated.
- Step 4** Click **Submit**. The system displays a confirmation screen.

Disassociating grid servers from caches

To disassociate a grid server from a cache, you must have write permission on the cache service (members of the Administrators group and the DataProviders group have this permission by default). In addition, you must have write permission on the grid server that you are disassociating from the cache service.

Follow these steps to disassociate a grid server from a cache:

- Step 1** Navigate to the View Grid Servers screen:

Home > Server management > View grid servers

View Grid Servers

You are viewing the grid servers in the current grid domain.

Server Name	View/Edit	Destroy Server	Attributes	Security
FRANCIUM.avaki.local	View/Edit	Destroy	Attributes	Security
cmagadiou.avaki.local	View/Edit	Destroy	Attributes	Security

- Step 2** Click the **View/Edit** link beside the grid server that you want to disassociate from a cache. The View/Edit Grid Server screen appears.

View/Edit Grid Server

To associate the grid server with a cache service, select the cache service from the pull-down list below. Once a cache is set, the grid server uses it to store pinned files and scheduled database operations so they can be accessed quickly.

Server name: magadiou.avaki.local

Current cache service: None

New cache service: ▼

- Step 3** From the “New cache service” pull-down menu, select **None**.
- Step 4** Click **Submit**. The system displays a confirmation screen.

Associating data grid access servers with caches

Follow these steps to associate a data grid access server with an external cache:

- Step 1** Navigate to the View DGASes screen:

Home > Server management > View DGASes

View DGASes

You are viewing the DGASes in the local grid domain.

Server Name	Configure	Cache	Mappings	Policy	Destroy	Attributes	Security
MyDGAS	Configure	Cache	Mappings	Policy	Destroy	Attributes	Security

- Step 2** Click the **Cache** link beside the data grid access server that you want to associate with a cache. The Manage DGAS Cache screen appears.

Manage DGAS Cache

Manage Internal Cache Service

<u>Clear credentials</u>	Clear all cached user credentials
<u>Delete from cache</u>	Delete a grid directory or file from DGAS cache
<u>Save cache</u>	Save a copy of the cache for a DGAS
<u>Sync file with proxy cache</u>	Force a DGAS to fetch grid directories or files from proxy cache if the cached version is newer than the version on the DGAS; the DGAS will copy the directory or file if it is not cached
<u>View/modify cache statistics</u>	View and reset all statistics counters for the DGAS cache

Manage External Cache Service

Set external cache service Associate an external proxy cache service with a DGAS

- Step 3** Click the **Set external cache service** link. The Set External Cache Service screen appears.

Set External Cache Service

DGAS name: MyDGAS

Current cache service: None

New cache service: ▼

- Step 4** From the “New cache service” pull-down menu, select the name of the external cache with which this data grid access server will be associated.

- Step 5** Click **Submit**.

Disassociating data grid access servers from caches

Follow these steps to associate a data grid access server from an external cache service:

Step 1 Navigate to the View DGASes screen:

Home > Server management > View DGASes

View DGASes

You are viewing the DGASes in the local grid domain.

Server Name	Configure	Cache	Mappings	Policy	Destroy	Attributes	Security
MyDGAS	Configure	Cache	Mappings	Policy	Destroy	Attributes	Security

Step 2 Click the **Cache** link beside the data grid access server that you want to disassociate from a cache. The Manage DGAS Cache screen appears.

Manage DGAS Cache

Manage Internal Cache Service

Clear credentials Clear all cached user credentials

Delete from cache Delete a grid directory or file from DGAS cache

Save cache Save a copy of the cache for a DGAS

Sync file with proxy cache Force a DGAS to fetch grid directories or files from proxy cache if the cached version is newer than the version on the DGAS; the DGAS will copy the directory or file if it is not cached

View/modify cache statistics View and reset all statistics counters for the DGAS cache

Manage External Cache Service

Set external cache service Associate an external proxy cache service with a DGAS

Step 3 Click the **Set external cache service** link. The Set External Cache Service screen appears.

Set External Cache Service

DGAS name: MyDGAS

Current cache service: None

New cache service:

Step 4 From the “New cache service” pull-down menu, select **None**.

Step 5 Click **Submit**.

Viewing and modifying cache service configuration

To display a list of cache services in the local grid domain, display configuration settings for cache services, or modify configuration settings, follow these steps.

Step 1 To view a list of the cache services in the local grid domain, navigate to the View Cache Services screen:

Home > Service management > View cache services

View Cache Services						
You are viewing the cache services in the local grid domain.						
Name	View/Edit	File Contents	DBOP Contents	Data Service Contents	Attributes	Security
MAGADIEUXP.sybase.com	View/Edit	File contents	DBOP contents	Data service contents	Attributes	Security
<input type="button" value="Done"/>						

The system displays a list of the cache services in the current Avaki domain.

Step 2 Click the **View/Edit** link beside the name of the cache service whose configuration you want to view or modify. The View/Edit Cache Service screen appears.

View/Edit Cache Service	
Cache service: blachmanxp.sybase.com	
Default coherence window (in seconds):	<input type="text" value="86400"/>
Default offline expiration (in seconds):	<input type="text" value="36288000"/>
Minimum warming interval (in seconds):	<input type="text" value="60"/>
Maximum size (in bytes):	<input type="text" value="8589934592"/>
Block size (in bytes):	<input type="text" value="1048576"/>
LRU purge threshold (% of maximum size):	<input type="text" value="0.96"/>
Current size (in bytes):	13
Free space on device (in bytes):	26733588480
Rejected block requests:	0
Cache evictions:	0
Cache misses:	1
Cache hits:	0
Ratio of misses to hits:	0:1
<input type="button" value="Submit"/> <input type="button" value="Cancel"/>	

- Step 3** Update the fields as desired, then click **Submit**. A confirmation screen appears when the update is complete.

The cache service settings and attributes are as follows:

Option	Description
Cache service	The name of the cache service.
Default coherence window	The duration (in seconds) during which the contents of the cached entry (such as a file) are assumed to be fresh after the cache service has last inspected the back-end source object for updates. The valid values are -1 and 0 to 2147483647 seconds. The default is 86400 (1 day). Set this attribute to -1 when the cache includes only static objects that should never expire.
Default offline expiration	The time (in seconds) for which content is allowed to remain valid after its source is determined to be offline. The valid values are -1 and 0 to 2147483647 seconds. The default is 36288000 (1 week). A value of -1 means that cache content is always available for offline access after the source is determined to be unreachable.
Minimum warming interval	Sets a lower boundary for the coherence windows and scheduling intervals of cached content. Coherence windows or scheduling intervals smaller than this interval will be rounded up to this value. The valid values are 0 to 2147483647 seconds. The default is 300 (5 minutes).
Maximum size	The maximum size (in bytes) of file content that the cache service can cache. The valid values are 1 to 2147483647 bytes. The default is 8589934592 bytes (8GB).
Block size	The size (in bytes) of the data blocks that cached files are broken into. For files that are cached on demand, the cache service caches only blocks that are needed by the clients that access the cache service. The default block size is 1MB (which is the default share server I/O protocol transfer block size).

Option	Description
LRU purge threshold	<p>A percentage (0.00 -1.00) that is applied to the maximum size to find the point at which the cache service should start removing least-recently-used (LRU) blocks to make room for new blocks. After the cache size reaches the purge threshold, the cache service makes a best effort to remove old blocks so that new blocks can be created as needed. The default purge threshold is 0.96; when the cache size reaches 7.68GB, the cache service starts evicting old cache blocks. The equilibrium point for the cache size is therefore 7.68GB. If enough requests are being handled (too many for the purging thread to keep up with), the cache service starts redirecting clients to the back-end source data when the hard limit of the maximum size (8GB) is reached.</p>
Current size	<p>The current size (in bytes) of the content that has been cached.</p>
Free space on device	<p>The amount of free disk space available on the storage device on which the cache service resides.</p>
Rejected block requests	<p>When a block request is rejected, the cache service has reached its maximum-size hard limit and is rejecting requests from cacheable items (such as cached files) to create new cache blocks.</p> <p>A rejected block request is an indication of the following:</p> <ul style="list-style-type: none"> • the maximum cache size value is too small; and • the load on the cache service is high enough that the best-effort, LRU block-purger inside cannot clear old blocks fast enough to keep up with the number of read requests that would like to generate new block files. <p>The result of a block rejection is that the client for which the cached file was attempting to create the block will be forwarded to the back-end source for that piece of information because the cache service can't cache the data its requesting.</p>
Evictions	<p>The number of cache blocks that have been evicted by the LRU-block-purger because of space limitations.</p>
Misses	<p>The number of times a user called for a file that is not in the cache.</p>

Option	Description
Hits	The number of times a user called for a file that is in the cache.
Ratio of misses to hits	The ratio of misses to hits.

Overriding cache service default settings

In addition to configuring the cache service default settings on a cachewide basis, you can configure some cache attributes on a per-file basis. If you set the following attributes on a per-file basis, the settings override the cachewide attributes that appear on the View/Edit Cache Service Screen ([page 116](#)):

- **cacheable/CoherenceWindow:** The duration (in seconds) during which the contents of the cached entry (such as a file) are assumed to be fresh after the cache service has last inspected the back-end source object for updates. The valid values are -1 and 0 to 2147483647 seconds. The default is 86400 (1 day). Set this attribute to -1 when the cache includes only static objects that should never expire.
- **cacheable/OfflineExpiration:** An integer specifying the time in seconds for which content is allowed to remain valid after its source is determined to be offline. The valid values are -1 and 0 to 2147483647 seconds. A value of -1 means that this file is always available for offline access when its source is determined to be unreachable.

For instructions on setting attributes, see the *Sybase Avaki EII Administration Guide*.

Managing caches

You can cache directories, files, database operation results, and data service results. When an object is cached, a copy of the object is maintained in the cache and refreshed according to a schedule or on demand. For more information about how on-demand and scheduled caching work, see [“On-demand caching” on page 108](#) and [“Scheduled caching” on page 109](#).

This section covers the following topics:

- [“Managing file or directory caches,”](#) below
- [“Managing database caches” on page 139](#)
- [“Managing data service caches” on page 152](#)

Managing file or directory caches

The following sections describe how to cache files or directories, how to mark a directory so that it does not get cached, how to view lists of marked files and directories, and how to unschedule and evict cached files or directories:

- [“Caching files or directories on a schedule” on page 121](#)
- [“Caching files or directories on demand” on page 130](#)
- [“Marking directories for no caching” on page 133](#)
- [“Viewing marked items” on page 134](#)
- [“Unscheduling/evicting files or directories” on page 136](#)
- [“Invalidating cached items” on page 136](#)

Caching files or directories on a schedule

Follow these steps to pin a file or directory for scheduled caching:









Step 1 Navigate to the Browse Directories screen:

Home > Data catalog management

Browse Directories

The top-level directory in a grid domain is /. Underneath is the /System subdirectory, which contains directories and files related to a particular grid domain, and the /Interconnects link to the Interconnects directory, which contains the root directories of any domains to which a domain is interconnected. A grid domain's administrator can view and modify the settings for an interconnected domain if the administrator has been granted access rights for modifying the domain.

You can add an Avaki directory at any level if you have write permission for the directory in which you want to create a directory. By default, only the Avaki administrator can create new directories at the top level, /.

<input type="checkbox"/>		/		Attributes	Security	
<input type="checkbox"/>		GeneratedViews		Attributes	Security	Categories
<input type="checkbox"/>		Interconnects		Attributes	Security	Categories
<input type="checkbox"/>		Metadata		Attributes	Security	Categories
<input type="checkbox"/>		Shares		Attributes	Security	Categories
<input type="checkbox"/>		System		Attributes	Security	Categories
<input type="checkbox"/>		WSDLs		Attributes	Security	Categories
<input type="checkbox"/>		Categories		Attributes	Security	Categories

Step 2 In the Browse Directories area, click on directory names to navigate to the directory or file you want to add to the cache service.

- Step 3** Click the check box to the left of the file name, then click **Update Cache**. The Cache Actions screen appears.

Cache Actions

You have chosen to perform a cache operation on the following objects in the grid directory **/Shares/MyAvakiShare**:

- project1.txt

Select the cache operation to perform:

Pin selected files

Pin selected directories only

Pin selected directories and all contents

Pin selected directories and subdirectories, and cache all contained files on demand

Cache selected files on demand

Mark selected directories for no caching

Evict selected items

Evict selected items and (if the items are directories) their contents

Invalidate selected items

Invalidate selected items and (if the items are directories) their contents

Cache service: ▼

- Step 4** Select which cache operation to perform:

- **Pin selected files:** Pin (mark) the selected files for scheduled caching.
- **Pin selected directories only:** Pin the selected directories for scheduled caching, but do not pin the contents of the directories.
- **Pin selected directories and all contents:** Recursively pin the directories and all their contents for caching. If you choose this option, the cache service actively caches the directories in the specified hierarchy and periodically inspects them for new files to be pinned. When the cache service reinspects the parent directory, it may repin a previously evicted file or subdirectory. If you want to prevent the cache service from repinning the file or directory, mark the item for no caching. See [“Marking directories for no caching”](#) on page 133.
- **Pin selected directories and subdirectories, and cache all contained files on demand:** Recursively tag the selected directories and their contents for on-demand caching. If you choose this option, the cache service actively caches the directories in the specified hierarchy and periodically inspects them for new files to be tagged for on-demand caching. When the cache service reinspects the parent directory, it may retag a previously evicted file or subdirectory. If you want to prevent the cache

service from retagging the file or directory, mark the item for no caching. See [“Marking directories for no caching”](#) on page 133.

Step 5 From the Cache service pull-down box, select the server that the cache is associated with.

Step 6 Click **Submit**. The Enter Caching Info screen appears.

Enter Caching Info

Choose an offline access configuration option for the given files. This setting determines how long a cached file will continue to be made available once its associated source file has gone offline. When you are finished, click **Submit** to define the schedules determining when the given files will be pinned.

Offline Access: File never expires when source is offline
 File is not available for offline access
 File observes cache service offline access default
 File is available for offline access for seconds

Step 7 (Optional) In the Offline Access section, specify how long a cached file or directory will continue to be made available once the back-end source file has gone offline. Choose one of the following options:

- File never expires when source is offline: The cached data will never expire when the back-end source is offline.
- File is not available for offline access: The cache service is not allowed to return cached data if the source is offline.
- File observes cache service offline access default: Use the cache service’s default offline access setting. By default, the cache service returns cached data for one week after the source goes offline.
- File is available for offline access for n seconds: Specify the interval (in seconds) before the file or directory expires from the cache.

Step 8 Click **Submit**. The Add New Schedule screen appears.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

One time
 Periodic
 Calendared
 Advanced

Starting Now
 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur every: minutes ▼

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Schedule exclusions:

No schedule exclusions are currently defined in this domain.

About schedule exclusions

A schedule exclusion is a named set of time periods. When an exclusion is applied to an event schedule, it prevents the event from occurring during the designated periods. To create a new schedule exclusion, click **Add Exclusion**, or click **View Exclusions** to see the list of exclusions currently defined in this domain. *Note: either choice will nullify any changes you've made to the current schedule.*

Step 9 Click a tab to choose the type of schedule: One time, Periodic, Calendared, or Advanced. The Advanced and Calendared options are similar. The Advanced option lets you use a cron expression—powerful but cryptic—to schedule the recurrence interval. The Calendared option offers a friendlier interface to a subset of the functionality enabled by cron expressions.

Step 10 Go to the appropriate procedure to complete your schedule entry:

- [“Configuring one-time refresh schedules” on page 125](#)
- [“Configuring calendared refresh schedules” on page 126](#)

- [“Configuring periodic or advanced refresh schedules” on page 128](#)

Configuring one-time refresh schedules. Follow the steps in [“Caching files or directories on a schedule” on page 121](#) before starting this subprocedure.

Add New Schedule

Select a time zone for this schedule:

Type of schedule:

One time
 Periodic
 Calendared
 Advanced

Do once: Now

:

Schedule exclusions:

No schedule exclusions are currently defined in this domain.

Follow these steps:

- Step 1** Choose a time zone for this schedule. The schedule can be specified according to the cache server’s time zone or relative to Greenwich Mean Time (GMT).
- Step 2** In the Do once field:
- If you want the one-time cache refresh to occur immediately, click the **Now** button.
 - If you want the cache refresh to occur later, click the lower button. Then use the pull-down menus to select the time, month, day and year.
- Step 3** Click **Submit** to save your schedule entry. The system displays the new entry on the Show Pin Schedules screen.

Note For instructions on setting up schedule exclusions—specific times when the cache is not updated according to the schedule—see [“Configuring schedule exclusions” on page 166](#).

Configuring calendared refresh schedules. Follow the steps in “Caching files or directories on a schedule” on page 121 before starting this subprocedure.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

Now
 After 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur at: 12 AM ▼ :00 ▼ on:

Days	Months	Years
<input checked="" type="radio"/> all <input type="radio"/> of week <input type="radio"/> of month <input type="radio"/> of week in month	Jan ▲ Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec all ▼	2004 ▲ 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 all ▼

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Schedule exclusions:

No schedule exclusions are currently defined in this domain.

Follow these steps:

- Step 1** Choose a time zone for this schedule. The schedule can be specified according to the cache server's time zone or relative to Greenwich Mean Time (GMT).
- Step 2** In the After field, specify when this schedule entry takes effect:

- If you want the one-time cache refresh to occur immediately, click the **Now** button.
 - If you want the cache refresh to occur later, click the lower button. Then use the pull-down menus to select the time, month, day and year.
- Step 3** In the **Recur at** field, use the pull-down menus to specify the time of day at which you want the cache refresh to take place. (If you want the refresh to occur more than once a day, you can use a periodic or advanced schedule, or you can create separate schedule entries for the other refreshes.)
- Step 4** In the **Days** column, choose how you want to specify days in this schedule entry:
- **All**: every day.
 - **Of week**: Sunday through Saturday—click one or more days.
 - **Of month**: 1, 2, 3...—click one or more days.
 - **Of week in month**: use the pull-down menus to choose the first, second, third, fourth, fifth, or last occurrence of any day of the week (the first Monday, for example).
- Step 5** In the **Months** column, select one or more months during which this schedule entry will be in effect, or select **all** for all months. Use Shift-click or Control-click to select multiple months.
- Step 6** In the **Years** column, select one or more years during which this schedule entry will be in effect, or select **all** for all years. Use Shift-click or Control-click to select multiple years.
- Step 7** In the **Continue recurring** field, specify how long you want this schedule entry to remain in effect: forever, for a specified number of refreshes, or until a specified date and time.
- Step 8** Click **Submit** to save your schedule entry. The system displays a summary of the new entry, including the time of next execution, on the **Show Pin Schedules** screen.

Note For instructions on setting up schedule exclusions—specific times when the cache is not refreshed according to the schedule—see [“Configuring schedule exclusions” on page 166](#).

Configuring periodic or advanced refresh schedules. Follow the steps in “[Caching files or directories on a schedule](#)” on page 121 before starting this subprocedure.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

One time
 Periodic
 Calendared
 Advanced

Now
 After 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur according to this cron expression:

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Schedule exclusions:

No schedule exclusions are currently defined in this domain.

One time
 Periodic
 Calendared
 Advanced

Now
 Starting 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur every: minutes ▼

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Follow these steps:

- Step 1** Choose a time zone for this schedule. The schedule can be specified according to the cache server's time zone or relative to Greenwich Mean Time (GMT).

- Step 2** In the After field, specify when this schedule entry takes effect:
- If you want the one-time cache refresh to occur immediately, click the **Now** button.
 - If you want the cache refresh to occur later, click the lower button. Then use the pull-down menus to select the time, month, day and year.
- Step 3** Use the Recur... field to specify the interval at which this schedule is executed:
- If you're creating a periodic schedule entry, enter an integer and select from the pull-down to specify an interval—for example, every 40 minutes, every 5 days, or every 2 months.
 - If you're creating an advanced schedule entry, you must enter a cron expression of this form in the Recur... field:


```
<seconds> <minutes> <hours> <days-of-month> <months>
<days-of-week> [<years>]
```

See the *Sybase Avaki EII Command Reference* for details of the cron syntax.
- Step 4** In the Continue recurring field, specify how long you want this schedule entry to remain in effect: forever, for a specified number of refreshes, or until a specified date and time.
- Step 5** Click **Submit** to save your schedule entry. The system displays a summary of the new entry, including the time of next execution, on the Show Pin Schedules screen.

Note For instructions on setting up schedule exclusions—specific times when the cache is not refreshed according to the schedule—see [“Configuring schedule exclusions” on page 166](#).

Caching files or directories on demand

Follow these steps to tag a file or directory for on-demand caching:









Step 1 Navigate to the Browse Directories screen:

Home > Data catalog management

Browse Directories

The top-level directory in a grid domain is /. Underneath is the /System subdirectory, which contains directories and files related to a particular grid domain, and the /Interconnects link to the Interconnects directory, which contains the root directories of any domains to which a domain is interconnected. A grid domain's administrator can view and modify the settings for an interconnected domain if the administrator has been granted access rights for modifying the domain.

You can add an Avaki directory at any level if you have write permission for the directory in which you want to create a directory. By default, only the Avaki administrator can create new directories at the top level, /.

<input type="checkbox"/>	 /	Attributes	Security	
<input type="checkbox"/>	 GeneratedViews	Attributes	Security	Categories
<input type="checkbox"/>	 Interconnects	Attributes	Security	Categories
<input type="checkbox"/>	 Metadata	Attributes	Security	Categories
<input type="checkbox"/>	 Shares	Attributes	Security	Categories
<input type="checkbox"/>	 System	Attributes	Security	Categories
<input type="checkbox"/>	 WSDLs	Attributes	Security	Categories
<input type="checkbox"/>	 Categories	Attributes	Security	Categories

Step 2 In the Browse Directories area, click on directory names to navigate to the directory or file you want to add to the cache service.

- Step 3** Click the check box to the left of the file name, then click **Update Cache**. The Cache Actions screen appears.

Cache Actions

You have chosen to perform a cache operation on the following objects in the grid directory **/Shares/MyAvakiShare**:

- project1.txt

Select the cache operation to perform:

- Pin selected files
- Pin selected directories only
- Pin selected directories and all contents
- Pin selected directories and subdirectories, and cache all contained files on demand
- Cache selected files on demand
- Mark selected directories for no caching
- Evict selected items
- Evict selected items and (if the items are directories) their contents
- Invalidate selected items
- Invalidate selected items and (if the items are directories) their contents

Cache service:

- Step 4** To tag the selected files for on-demand caching, select the option “Cache selected files on demand.”
- Step 5** From the Cache service pull-down box, select the server that the cache is associated with.
- Step 6** Click **Submit**. The Enter Caching Info screen appears.

Enter Caching Info

Choose an offline access configuration option for the given files. This setting determines how long a cached file will continue to be made available once its associated source file has gone offline. Also choose a coherence window configuration option, specifying how long the cache service will assume the cached file contents are fresh. When you are finished, click **Submit**.

Offline Access: File never expires when source is offline
 File is not available for offline access
 File observes cache service offline access default
 File is available for offline access for seconds

Coherence Window: File never expires
 File observes cache service coherence window default
 File has a coherence window of seconds

Step 7 (Optional) In the Offline Access section, specify how long a cached file or directory will continue to be made available once the back-end source file has gone offline. Choose one of the following options:

- File never expires when source is offline: The cached data will never expire when the back-end source is offline.
- File is not available for offline access: The cache service is not allowed to return cached data if the source is offline.
- File observes cache service offline access default: Use the cache service's default offline access setting. By default, the cache service returns cached data for one week after the source goes offline.
- File is available for offline access for n seconds: Specify the interval (in seconds) before the file or directory expires from the cache.

Step 8 (Optional) In the Coherence Window section, specify how long the cache service will assume that the cached file contents are fresh. Choose one of the following options:

- File never expires: The cached data never expires.
- File observes cache service coherence window default: Use the cache service's default coherence window setting. By default, the cache service returns cached data for one day.
- File has a coherence window of n seconds: Specify the interval (in seconds) before the file expires from the cache.

Step 9 Click **Submit**. The system displays a confirmation screen.

Marking directories for no caching

If you mark a large directory hierarchy for scheduled or on-demand caching, you can mark certain subdirectories so that they will not be cached. When a subdirectory is marked for no caching, the directory and its contents will not be cached when the parent hierarchy is cached.

To mark a directory for no caching, do the following:









Step 1 Navigate to the Browse Directories screen:

Home > Data catalog management

Browse Directories

The top-level directory in a grid domain is /. Underneath is the /System subdirectory, which contains directories and files related to a particular grid domain, and the /Interconnects link to the Interconnects directory, which contains the root directories of any domains to which a domain is interconnected. A grid domain's administrator can view and modify the settings for an interconnected domain if the administrator has been granted access rights for modifying the domain.

You can add an Avaki directory at any level if you have write permission for the directory in which you want to create a directory. By default, only the Avaki administrator can create new directories at the top level, /.

<input type="checkbox"/>	 /	Attributes	Security	
<input type="checkbox"/>	 GeneratedViews	Attributes	Security	Categories
<input type="checkbox"/>	 Interconnects	Attributes	Security	Categories
<input type="checkbox"/>	 Metadata	Attributes	Security	Categories
<input type="checkbox"/>	 Shares	Attributes	Security	Categories
<input type="checkbox"/>	 System	Attributes	Security	Categories
<input type="checkbox"/>	 WSDLs	Attributes	Security	Categories
<input type="checkbox"/>	 Categories	Attributes	Security	Categories

Step 2 In the Browse Directories area, click on directory names to navigate to the directory you want to mark for no caching.

- Step 3** Click the check box to the left of the directory name, then click **Update Cache**. The Cache Actions screen appears.

Cache Actions

You have chosen to perform a cache operation on the following objects in the grid directory **/Shares/MyAvakiShare**:

- project1.txt

Select the cache operation to perform:

Pin selected files

Pin selected directories only

Pin selected directories and all contents

Pin selected directories and subdirectories, and cache all contained files on demand

Cache selected files on demand

Mark selected directories for no caching

Evict selected items

Evict selected items and (if the items are directories) their contents

Invalidate selected items

Invalidate selected items and (if the items are directories) their contents

Cache service: ▼

- Step 4** Select the button labeled “Mark selected directories for no caching.”
- Step 5** From the Cache service pull-down box, select the server that the cache is associated with.
- Step 6** Click **Submit**. The system displays a confirmation screen.

Viewing marked items

To view a list of the files or directories that are marked (pinned) for caching in a cache service, navigate to the View Cache Services screen:

Home > Service management > View cache services

View Cache Services

You are viewing the cache services in the local grid domain.

Name	View/Edit	File Contents	DBOP Contents	Data Service Contents	Attributes	Security
MAGADIEUXP.sybase.com	View/Edit	File contents	DBOP contents	Data service contents	Attributes	Security

Click **File contents** beside the cache service whose contents you want to view. The system displays a list of items that are pinned in the cache service.

View Cached Files	
You are viewing the files pinned in the cache service on blachmanxp.sybase.com .	
Select	File Name
<input type="checkbox"/>	[Show Info] /Shares/MyAvakiShare/MyFile1.txt
<input type="button" value="Check All"/> <input type="button" value="Clear All"/>	
<input type="button" value="Evict All Checked"/> <input type="button" value="Done"/>	

To see more information about any given item, click **Show Info** next to the item:

View Cached Files	
You are viewing the files pinned in the cache service on blachmanxp.sybase.com .	
Select	File Name
<input type="checkbox"/>	[Hide Info] /Shares/MyAvakiShare/MyFile1.txt <hr/> Actively Cached Schedules: Every 60 minutes, next execution Thu Jun 2 3:20:58 PM EDT 2005 Offline Access: Observes Cache Service Default Valid: true Metadata Last Checked: Jun 2, 2005 2:21:02 PM Metadata Last Updated: Jun 2, 2005 2:21:02 PM
<input type="button" value="Check All"/> <input type="button" value="Clear All"/>	
<input type="button" value="Evict All Checked"/> <input type="button" value="Done"/>	

To restore the initial view (that is, to hide that extra information), click **Hide Info** next to the item.

Unscheduler/evicting files or directories

When you unschedule or evict a file or directory, the item is purged from the cache service and unscheduled if it's pinned.

To unschedule or evict a file or directory from the cache, do the following:

- Step 1** Navigate to the View Cache Services screen:

Home > Service management > View cache services

View Cache Services						
You are viewing the cache services in the local grid domain.						
Name	View/Edit	File Contents	DBOP Contents	Data Service Contents	Attributes	Security
MAGADIEUXP.sybase.com	View/Edit	File contents	DBOP contents	Data service contents	Attributes	Security
<input type="button" value="Done"/>						

- Step 2** Click **File contents** beside the cache service whose contents you want to unschedule or evict. The system displays a list of items that are in the cache service.

View Cached Files	
You are viewing the files pinned in the cache service on blachmanxp.sybase.com.	
Select	File Name
<input type="checkbox"/>	[Show Info] /Shares/MyAvakiShare/MyFile1.txt
<input type="button" value="Check All"/> <input type="button" value="Clear All"/>	
<input type="button" value="Evict All Checked"/> <input type="button" value="Done"/>	

- Step 3** Click boxes in the Select column to choose the items you want to unschedule or evict.

- Step 4** Click **Evict All Checked**. If the file or directory is scheduled, the system unschedules it. If the file or directory is cached on demand, the system evicts it.

Invalidating cached items

When you mark a file or directory in a cache as invalid, the cached copy of the object will no longer be used when a user requests the object; the object will be refreshed in the cache the next time it is accessed or upon its next scheduled refresh.

To mark a file or directory in a cache as invalid, do the following:

Step 1 Navigate to the Browse Directories screen:

Home > Data catalog management

Browse Directories

The top-level directory in a grid domain is /. Underneath is the /System subdirectory, which contains directories and files related to a particular grid domain, and the /Interconnects link to the Interconnects directory, which contains the root directories of any domains to which a domain is interconnected. A grid domain's administrator can view and modify the settings for an interconnected domain if the administrator has been granted access rights for modifying the domain.

You can add an Avaki directory at any level if you have write permission for the directory in which you want to create a directory. By default, only the Avaki administrator can create new directories at the top level, /.

<input type="checkbox"/>	/	Attributes	Security	
<input type="checkbox"/>	GeneratedViews	Attributes	Security	Categories
<input type="checkbox"/>	Interconnects	Attributes	Security	Categories
<input type="checkbox"/>	Metadata	Attributes	Security	Categories
<input type="checkbox"/>	Shares	Attributes	Security	Categories
<input type="checkbox"/>	System	Attributes	Security	Categories
<input type="checkbox"/>	WSDLs	Attributes	Security	Categories
<input type="checkbox"/>	Categories	Attributes	Security	Categories

Step 2 In the Browse Directories area, click on directory names to navigate to the directory or file you want to remove from the cache service.

- Step 3** Click the check box to the left of the directory or file name, then click **Update Cache**. The Cache Actions screen appears.

Cache Actions

You have chosen to perform a cache operation on the following objects in the grid directory **/Shares/MyAvakiShare**:

- project1.txt

Select the cache operation to perform:

Pin selected files

Pin selected directories only

Pin selected directories and all contents

Pin selected directories and subdirectories, and cache all contained files on demand

Cache selected files on demand

Mark selected directories for no caching

Evict selected items

Evict selected items and (if the items are directories) their contents

Invalidate selected items

Invalidate selected items and (if the items are directories) their contents

Cache service: ▼

- Step 4** Select which cache operation to perform:

- **Invalidate selected items:** Flag the selected files or directories (but not the directories' contents) as invalid. When a cached copy of an object is invalidated, the cached data will no longer be used when a user requests the object; the object will be refreshed in the cache the next time it is accessed or upon its next scheduled refresh.
- **Invalidate selected items and (if the items are directories) their contents:** Flag the selected files or directories (including the directories' contents) as invalid. When a cached copy of an object is invalidated, the cached data will no longer be used when a user requests the object; the object will be refreshed in the cache the next time it is accessed or upon its next scheduled refresh.

- Step 5** From the Cache service pull-down box, select the server that the cache is associated with.

- Step 6** Click **Submit**. The system displays a confirmation screen.

Note If you invalidate an object that is pinned, the object will be refreshed in the cache the next time it is accessed or upon its next scheduled refresh.

Managing database caches

This section covers the following topics:

- “Caching database operations on a schedule,” below
- “Caching database operations on demand” on page 147
- “Viewing cached database operations” on page 149
- “Unscheduler/evicting database operations” on page 151

Caching database operations on a schedule

When you schedule a database operation or a virtual database operation, you can define a specific calendar-based interval at which a cache entry should be refreshed.

Follow these steps to schedule a database operation or a virtual database operation:

Step 1 To schedule...

- A database operation, navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations

Grid domain: ▼

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

The system displays a list of the database operations in the current domain.

- A virtual database operation, navigate to the View Virtual Database Operations screen:

Home > Data integration > Manage virtual database operations

View Virtual Database Operations

Grid domain:

You are viewing the virtual database operations in the current domain.

Name	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyVirtualDBOperation	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

The system displays a list of the database operations in the current domain.

- Step 2** Click the **Schedule** link beside the database operation or virtual database operation you want to schedule. The Schedule Database Operation screen appears.

Schedule Database Operation

You have chosen to schedule the database operation **MyDBOperation** for periodic update in a cache service. You can choose to do this with any cache service in this domain. If you do it on the server local to the operation (shown with an asterisk * before its name), your scheduling will affect all requests using your indicated set of parameter values throughout the domain. If you schedule your updates on any other server, they will affect only requests made from that server. Next, specify the values of the input parameters for the operation (if there are any). Finally, click **Add Schedule** to go on to define the schedule on which your updates will occur, or **View Schedules** to review (and manage) any existing schedules for this combination of database operation, parameter values and server.

Cache service:

SQL statement: select * from emp where ename = ?

Parameter 1: VARCHAR

- Step 3** Select the cache service to use. If you want to reduce the load on the underlying database, select the server that is local to the operation (the server with an asterisk * before its name) as the cache service. If you want to cache the results on a remote grid server (and thereby reduce network traffic when requests are made from that server), select the cache service for the related grid server.
- Step 4** Specify the values of the input parameters for the operation (if there are any).
- Step 5** Click **Add Schedule**.

Step 6 The Add New Schedule screen appears.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

Starting Now
 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur every: minutes ▼

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Schedule exclusions:

No schedule exclusions are currently defined in this domain.

About schedule exclusions

A schedule exclusion is a named set of time periods. When an exclusion is applied to an event schedule, it prevents the event from occurring during the designated periods. To create a new schedule exclusion, click **Add Exclusion**, or click **View Exclusions** to see the list of exclusions currently defined in this domain. *Note: either choice will nullify any changes you've made to the current schedule.*

Step 7 Click a tab to choose the type of schedule: One time, Periodic, Calendared, or Advanced. The Advanced and Calendared options are similar. The Advanced option lets you use a cron expression—powerful but cryptic—to schedule the recurrence interval. The Calendared option offers a friendlier interface to a subset of the functionality enabled by cron expressions.

Step 8 Go to the appropriate procedure to complete your schedule entry:

- [“Configuring one-time refresh schedules” on page 125](#)
- [“Configuring calendared refresh schedules” on page 126](#)

- [“Configuring periodic or advanced refresh schedules” on page 128](#)

Configuring one-time refresh schedules. Follow the steps in [“Caching database operations on a schedule” on page 139](#) before starting this subprocedure.

Add New Schedule

Select a time zone for this schedule:

Type of schedule:

One time
 Periodic
 Calendared
 Advanced

Do once: Now

:

Follow these steps:

- Step 1** Choose a time zone for this schedule. The schedule can be specified according to the cache server’s time zone or relative to Greenwich Mean Time (GMT).
- Step 2** In the Do once field:
- If you want the one-time cache refresh to occur immediately, click the **Now** button.
 - If you want the cache refresh to occur later, click the lower button. Then use the pull-down menus to select the time, month, day and year.
- Step 3** Click **Submit** to save your schedule entry. The system displays the new entry on the Show Database Operations screen.

Note For instructions on setting up schedule exclusions—specific times when the cache is not updated according to the schedule—see [“Configuring schedule exclusions” on page 166](#).

Configuring calendared refresh schedules. Follow the steps in “[Caching database operations on a schedule](#)” on page 139 before starting this subprocedure.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

One time
 Periodic
 Calendared
 Advanced

Now
 After 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur at: 12 AM ▼ :00 ▼ on:

Days	Months	Years
<input checked="" type="radio"/> all <input type="radio"/> of week <input type="radio"/> of month <input type="radio"/> of week in month	Jan ▲ Feb ▲ Mar ▲ Apr ▲ May ▲ Jun ▲ Jul ▲ Aug ▲ Sep ▲ Oct ▲ Nov ▲ Dec ▲ all ▼	2004 ▲ 2005 ▲ 2006 ▲ 2007 ▲ 2008 ▲ 2009 ▲ 2010 ▲ 2011 ▲ 2012 ▲ 2013 ▲ 2014 ▲ 2015 ▲ all ▼

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Follow these steps:

- Step 1** Choose a time zone for this schedule. The schedule can be specified according to the cache server’s time zone or relative to Greenwich Mean Time (GMT).
- Step 2** In the After field, specify when this schedule entry takes effect:
- If you want the one-time cache refresh to occur immediately, click the **Now** button.
 - If you want the cache refresh to occur later, click the lower button. Then use the pull-down menus to select the time, month, day and year.

- Step 3** In the **Recur at** field, use the pull-down menus to specify the time of day at which you want the cache refresh to take place. (If you want the refresh to occur more than once a day, you can use a periodic or advanced schedule, or you can create separate schedule entries for the other refreshes.)
- Step 4** In the **Days** column, choose how you want to specify days in this schedule entry:
- **All:** every day.
 - **Of week:** Sunday through Saturday—click one or more days.
 - **Of month:** 1, 2, 3...—click one or more days.
 - **Of week in month:** use the pull-down menus to choose the first, second, third, fourth, fifth, or last occurrence of any day of the week (the first Monday, for example).
- Step 5** In the **Months** column, select one or more months during which this schedule entry will be in effect, or select **all** for all months. Use Shift-click or Control-click to select multiple months.
- Step 6** In the **Years** column, select one or more years during which this schedule entry will be in effect, or select **all** for all years. Use Shift-click or Control-click to select multiple years.
- Step 7** In the **Continue recurring** field, specify how long you want this schedule entry to remain in effect: forever, for a specified number of refreshes, or until a specified date and time.
- Step 8** Click **Submit** to save your schedule entry. The system displays a summary of the new entry, including the time of next execution, on the **Show Database Operations** screen.

Note For instructions on setting up schedule exclusions—specific times when the cache is not refreshed according to the schedule—see [“Configuring schedule exclusions” on page 166](#).

Configuring periodic or advanced refresh schedules. Follow the steps in “[Caching database operations on a schedule](#)” on page 139 before starting this subprocedure.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

Now
 After 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur according to this cron expression:

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Now
 Starting 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur every: minutes ▼

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Follow these steps:

- Step 1** Choose a time zone for this schedule. The schedule can be specified according to the cache server’s time zone or relative to Greenwich Mean Time (GMT).
- Step 2** In the After field, specify when this schedule entry takes effect:
- If you want the one-time cache refresh to occur immediately, click the **Now** button.
 - If you want the cache refresh to occur later, click the lower button. Then use the pull-down menus to select the time, month, day and year.

- Step 3** Use the `Recur...` field to specify the interval at which this schedule is executed:
- If you're creating a periodic schedule entry, enter an integer and select from the pull-down to specify an interval—for example, every 40 minutes, every 5 days, or every 2 months.
 - If you're creating an advanced schedule entry, you must enter a cron expression of this form in the `Recur...` field:

```
<seconds> <minutes> <hours> <days-of-month> <months>  
<days-of-week> [<years>]
```

See the *Sybase Avaki EII Command Reference* for details of the cron syntax.

- Step 4** In the `Continue recurring` field, specify how long you want this schedule entry to remain in effect: forever, for a specified number of refreshes, or until a specified date and time.
- Step 5** Click **Submit** to save your schedule entry. The system displays a summary of the new entry, including the time of next execution, on the `Show Database Operations` screen.

Note For instructions on setting up schedule exclusions—specific times when the cache is not refreshed according to the schedule—see [“Configuring schedule exclusions” on page 166](#).

Caching database operations on demand

Follow these steps to tag database operations or virtual database operations for on-demand caching:

Step 1 To tag...

- A database operation for on-demand caching, navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations

Grid domain:

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

The system displays a list of the database operations in the current domain.

- A virtual database operation for on-demand caching, navigate to the View Virtual Database Operations screen:

Home > Data integration > Manage virtual database operations

View Virtual Database Operations

Grid domain:

You are viewing the virtual database operations in the current domain.

Name	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyVirtualDBOperation	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

The system displays a list of the virtual database operations in the current domain.

- Step 2** Click the **View/Edit** link beside the name of the database operation or virtual database operation to cache on demand. The Update Database Operation screen appears, showing the details for the database operation or virtual database operation.

Update Virtual Database Operation

Fill in the information about your virtual database operation below. (If there's a file in your domain that contains your SQL statement, click **Browse** to import it.) If your SQL statement has one or more parameters, you will be asked to specify additional information about those parameters when you click **Continue**.

Database schema:

Server: MAGADIEUXP.sybase.com

Logical name: MyVirtualDBOP

Description (optional):

Metadata: [Click here](#) to generate or view the schema for this virtual database operation, generate or remove it as a virtual SQL view, or browse its dependencies.

SQL statement:

Cached data expiration: No caching
 Never expires
 Expires after seconds

Run database operation as: The user running the operation
 A specific user: [Browse](#)
(e.g. userName@authService.authServiceType.domain)

- Step 3** Fill in the form:

- **Cached data expiration:** Select one of the following to indicate whether the data is cached and, if so, when the data expires from the cache:
 - **No caching:** The data is not cached.
 - **Never expires:** The data never expires from the cache.
 - **Expires after *n* seconds:** Specify the interval (in seconds) before the data expires from the cache.

- Step 4** Click **Submit**. The system displays a confirmation screen.

Viewing cached database operations

To display a list of the database operations and virtual database operations in a cache service, do the following:

Step 1 Navigate to the View Cache Services screen:

Home > Service management > View cache services

View Cache Services

You are viewing the cache services in the local grid domain.

Name	View/Edit	File Contents	DBOP Contents	Data Service Contents	Attributes	Security
MAGADIEUXP.sybase.com	View/Edit	File contents	DBOP contents	Data service contents	Attributes	Security

The system displays a list of the cache services in the current domain.

Step 2 Click the **DBOP Contents** link beside the name of the cache service whose database operations you want to view. The View Database Cache screen appears, listing the database operations in the cache service. The example below shows the four possible combinations of caching and invalidation conditions:

- **CherylDomain.MyDBConnector.MyDBOperation1:** This database operation is scheduled to be executed and its results have been cached. You can invalidate the current cached results or unschedule the cache schedule.
- **CherylDomain.MyDBConnector.MyDBOperation2:** The database operation result is cached on demand. You can invalidate the cached result.
- **CherylDomain.MyDBConnector.MyDBOperation3:** The cache service contains an entry for the database operation and its cached metadata, but the cache service contains no cached results for the database operation. This situation occurs if the database operation's result is cached and then later invalidated.

- **CherylDomain.MyDBConnector.MyDBOperation4:** The cache service contains an entry for the database operation and its metadata, but the database operation results will not be cached.

View Database Cache

You are viewing the database operations in the cache service on **blachmanxp.sybase.com**. To manage the caching of an entire database operation, place a check mark beside the database operation and click one of the following buttons:

- **Evict All Checked** untags, unschedules, and purges all cached result sets and metadata for the specified database operations.
- **Invalidate All Checked** invalidates all of the cached result sets and metadata for the specified database operations.

You can manage individual database operation result sets by clicking the embedded **Invalidate** or **Unschedule** links to invalidate a result set or unschedule a particular cache schedule.

Database Operation	Parameters	Status
<input type="checkbox"/> CherylDomain.MyDBConnector.MyDBOperation1	No Parameters [Invalidate]	Scheduled: Every 1 days, next execution Fri Jun 3 10:33:57 AM EDT 2005 [Unschedule]
<input type="checkbox"/> CherylDomain.MyDBConnector.MyDBOperation2	No Parameters [Invalidate]	Cached On-Demand
<input type="checkbox"/> CherylDomain.MyDBConnector.MyDBOperation3	Not Applicable	Cached On-Demand (Metadata Only)
<input type="checkbox"/> CherylDomain.MyDBConnector.MyDBOperation4	Not Applicable	Data Caching Not Permitted (Metadata Cached)

Check All
Clear All

Evict All Checked
Invalidate All Checked
Done

Unscheduler/evicting database operations

When you unschedule or evict a database operation or virtual database operation, the database operation is purged from the cache service and unscheduled if it's pinned.

To unschedule or evict a scheduled database operation or virtual database operation, do the following:

Step 1 Navigate to the View Cache Services screen:

Home > Service management > View cache services

View Cache Services

You are viewing the cache services in the local grid domain.

Name	View/Edit	File Contents	DBOP Contents	Data Service Contents	Attributes	Security
MAGADIEUXP.sybase.com	View/Edit	File contents	DBOP contents	Data service contents	Attributes	Security

The system displays a list of the cache services in the current domain.

Step 2 Click the **DBOP Contents** link beside the name of the cache service that you want to unschedule or evict. The View Database Cache screen appears.

View Database Cache

You are viewing the database operations in the cache service on **blachmanxp.sybase.com**. To manage the caching of an entire database operation, place a check mark beside the database operation and click one of the following buttons:

- **Evict All Checked** untags, unschedules, and purges all cached result sets and metadata for the specified database operations.
- **Invalidate All Checked** invalidates all of the cached result sets and metadata for the specified database operations.

You can manage individual database operation result sets by clicking the embedded **Invalidate** or **Unschedule** links to invalidate a result set or unschedule a particular cache schedule.

Database Operation	Parameters	Status
<input type="checkbox"/> CherylDomain.MyDBConnector.MyDBOperation	VARCHAR: "pfein" [Invalidate]	Scheduled: Every 60 minutes, next execution Thu Jun 2 11:50:43 AM EDT 2005 [Unschedule]

- Step 3** (Optional) Click boxes in the Query Name column to choose the database operations you want to unschedule or evict, then select one of the following:
- To unpin, unschedule, and purge all cached results and metadata for the specified database operation, click **Evict All Checked**.
 - To invalidate all of the cached results data and metadata for the specified database operations, click **Invalidate All Checked**.
- Step 4** (Optional) To invalidate a specific cached result set, click the **Invalidate** link in the Parameters entry for that result set.
- Step 5** (Optional) To unschedule a specific schedule, click the **Unschedule** link beside the description of that schedule.

Managing data service caches

This section covers the following topics:

- [“Caching data services on a schedule,”](#) below
- [“Caching data services on demand”](#) on page 160
- [“Viewing data services”](#) on page 163
- [“Unscheduling/evicting data services”](#) on page 165

Caching data services on a schedule

When you schedule a data service, you can define a specific calendar-based interval at which a cache entry should be refreshed.

Follow these steps to schedule a data service:

- Step 1** Navigate to the View Data Services screen:

Home > Data integration > Manage data services

View Data Services

Grid domain:

You are viewing the data services in the current domain.

Name	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata
MyDataService	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata

The system displays a list of the data services in the current domain.

- Step 2** Click the **Schedule** link beside the data service you want to schedule. The Schedule Data Service screen appears.

Schedule Data Service

You have chosen to schedule the data service **MyDataService** for periodic update in a cache service. You can choose to do this with any cache service in this domain. If you do it on the server local to the data service (shown with an asterisk * before its name), your scheduling will affect all requests using your indicated set of parameter values throughout the domain. If you schedule your updates on any other server, they will affect only requests made from that server. Next, specify the values of the input parameters for the service (if there are any). Finally, click **Add Schedule** to go on to define the schedule on which your updates will occur, or **View Schedules** to review (and manage) any existing schedules for this combination of data service, parameter values and server.

Cache service: ▼

Parameters: This data service does not require any parameters.

- Step 3** Select the cache service to use. If you want to reduce load on the underlying database, select the server that is local to the operation (the server with an asterisk * before its name) as the cache service. If you want to cache the results on a remote grid server (and thereby reduce network traffic when requests are made from that server), select the cache service for the related grid server.

- Step 4** Click **Add Schedule**.

Step 5 The Add New Schedule screen appears.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

One time
 Periodic
 Calendared
 Advanced

Starting Now
 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur every: minutes ▼

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Schedule exclusions:

No schedule exclusions are currently defined in this domain.

About schedule exclusions

A schedule exclusion is a named set of time periods. When an exclusion is applied to an event schedule, it prevents the event from occurring during the designated periods. To create a new schedule exclusion, click **Add Exclusion**, or click **View Exclusions** to see the list of exclusions currently defined in this domain. *Note: either choice will nullify any changes you've made to the current schedule.*

Step 6 Click a tab to choose the type of schedule: One time, Periodic, Calendared, or Advanced. The Advanced and Calendared options are similar. The Advanced option lets you use a cron expression—powerful but cryptic—to schedule the recurrence interval. The Calendared option offers a friendlier interface to a subset of the functionality enabled by cron expressions.

Step 7 Go to the appropriate procedure to complete your schedule entry:

- [“Configuring one-time refresh schedules,”](#) below
- [“Configuring calendared refresh schedules”](#) on page 126

- [“Configuring periodic or advanced refresh schedules” on page 128](#)

Configuring one-time refresh schedules. Follow the steps in [“Caching data services on a schedule” on page 152](#) before starting this subprocedure.

Add New Schedule

Select a time zone for this schedule:

Type of schedule:

One time
 Periodic
 Calendared
 Advanced

Do once: Now

:

Follow these steps:

- Step 1** Choose a time zone for this schedule. The schedule can be specified according to the cache server’s time zone or relative to Greenwich Mean Time (GMT).
- Step 2** In the Do once field:
- If you want the one-time cache refresh to occur immediately, click the **Now** button.
 - If you want the cache refresh to occur later, click the lower button. Then use the pull-down menus to select the time, month, day and year.
- Step 3** Click **Submit** to save your schedule entry. The system displays the new entry on the Show Data Service Schedules screen.

Note For instructions on setting up schedule exclusions—specific times when the cache is not updated according to the schedule—see [“Configuring schedule exclusions” on page 166](#).

Configuring calendared refresh schedules. Follow the steps in “Caching data services on a schedule” on page 152 before starting this subprocedure.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

Now
 After 12 AM :00 Jan 1 2005

Recur at: 12 AM :00 on:

Days	Months	Years
<input checked="" type="radio"/> all <input type="radio"/> of week <input type="radio"/> of month <input type="radio"/> of week in month	Jan ▲ Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec all ▼	2004 ▲ 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 all ▼

Continue recurring

forever
 at most times
 until 12 AM :00 Jan 1 2005

Follow these steps:

- Step 1** Choose a time zone for this schedule. The schedule can be specified according to the cache server’s time zone or relative to Greenwich Mean Time (GMT).
- Step 2** In the After field, specify when this schedule entry takes effect:
- If you want the one-time cache refresh to occur immediately, click the **Now** button.
 - If you want the cache refresh to occur later, click the lower button. Then use the pull-down menus to select the time, month, day and year.

- Step 3** In the **Recur at** field, use the pull-down menus to specify the time of day at which you want the cache refresh to take place. (If you want the refresh to occur more than once a day, you can use a periodic or advanced schedule, or you can create separate schedule entries for the other refreshes.)
- Step 4** In the **Days** column, choose how you want to specify days in this schedule entry:
- **All**: every day.
 - **Of week**: Sunday through Saturday—click one or more days.
 - **Of month**: 1, 2, 3...—click one or more days.
 - **Of week in month**: use the pull-down menus to choose the first, second, third, fourth, fifth, or last occurrence of any day of the week (the first Monday, for example).
- Step 5** In the **Months** column, select one or more months during which this schedule entry will be in effect, or select **all** for all months. Use Shift-click or Control-click to select multiple months.
- Step 6** In the **Years** column, select one or more years during which this schedule entry will be in effect, or select **all** for all years. Use Shift-click or Control-click to select multiple years.
- Step 7** In the **Continue recurring** field, specify how long you want this schedule entry to remain in effect: forever, for a specified number of refreshes, or until a specified date and time.
- Step 8** Click **Submit** to save your schedule entry. The system displays a summary of the new entry, including the time of next execution, on the **Show Data Service Schedules** screen.

Note For instructions on setting up schedule exclusions—specific times when the cache is not refreshed according to the schedule—see [“Configuring schedule exclusions” on page 166](#).

Configuring periodic or advanced refresh schedules. Follow the steps in “[Caching data services on a schedule](#)” on page 152 before starting this subprocedure.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

Now
 After 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur according to this cron expression:

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Schedule exclusions:

No schedule exclusions are currently defined in this domain.

Starting Now
 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur every: minutes ▼

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Follow these steps:

- Step 1** Choose a time zone for this schedule. The schedule can be specified according to the cache server's time zone or relative to Greenwich Mean Time (GMT).

- Step 2** In the After field, specify when this schedule entry takes effect:
- If you want the one-time cache refresh to occur immediately, click the **Now** button.
 - If you want the cache refresh to occur later, click the lower button. Then use the pull-down menus to select the time, month, day and year.
- Step 3** Use the Recur... field to specify the interval at which this schedule is executed:
- If you're creating a periodic schedule entry, enter an integer and select from the pull-down to specify an interval—for example, every 40 minutes, every 5 days, or every 2 months.
 - If you're creating an advanced schedule entry, you must enter a cron expression of this form in the Recur... field:


```
<seconds> <minutes> <hours> <days-of-month> <months>
<days-of-week> [<years>]
```

See the *Sybase Avaki EII Command Reference* for details of the cron syntax.
- Step 4** In the Continue recurring field, specify how long you want this schedule entry to remain in effect: forever, for a specified number of refreshes, or until a specified date and time.
- Step 5** Click **Submit** to save your schedule entry. The system displays a summary of the new entry, including the time of next execution, on the Show Data Service Schedules screen.

Note For instructions on setting up schedule exclusions—specific times when the cache is not refreshed according to the schedule—see [“Configuring schedule exclusions” on page 166](#).

Caching data services on demand

Follow these steps to tag data services for on-demand caching:

Step 1 Navigate to the View Data Services screen:

Home > Data integration > Manage data services

View Data Services

Grid domain: ▼

You are viewing the data services in the current domain.

Name	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata
MyDataService	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata

The system displays a list of the data services in the current domain.

- Step 2** Click the **View/Edit** link beside the name of the data service to cache on demand. The Update Data Services screen appears, showing the details for the data service.

Update Data Service

No Parameters

Input Stream

Name: Primary Input
Stream Type: XML
From: File
Target: /Shares/MyTestShare2/test-xsl/test.xml
[Edit Stream](#)

Input Stream

Name: Secondary Inputs
Stream Type: XML[]
Instances: 0 [Add](#)

Plug-in

Name: to-html.xsl
Description: XSLT transform using Saxon
Coherence Window: 5 minutes
[Change Plug-in](#)

Input Parameter

Name: HeadCellColour
Type: VARCHAR
Description: XSLT Stylesheet parameter
Value: #D2D2D2
[Edit Parameter](#)

Input Parameter

Name: Title
Type: VARCHAR
Description: XSLT Stylesheet parameter
Value: Vacations
[Edit Parameter](#)

Input Parameter

Name: CellColour
Type: VARCHAR
Description: XSLT Stylesheet parameter
Value: #E2E2E2
[Edit Parameter](#)

Output Stream

Name:

Step 3 Click **Next**. The second Update Data Services screen appears.

Update Data Service

You may change the value for the expiration of the data service's data and/or the coherence window for the data service's plug-in. When you are done, click **Next** to update the data service.

Or click **Return to Overview** to return to editing the parameters, input stream(s), plug-in or output type of this data service.

Data service name: MyDataService

Description:

Cached data expiration: No caching
 Never expires
 Expires after

Plug-in coherence window:

Execute data service as: The user calling the data service
 A specific user:
(e.g. `userName@authService.authServiceType.domain`)

- In the Cached data expiration field, select one of the following to indicate whether the data is cached and, if so, when the data expires from the cache:
 - No caching: The data is not cached.
 - Never expires: The data never expires from the cache.
 - Expires after *n* seconds: Specify the interval (in seconds, minutes, hours, or days) before the data expires from the cache.
- In the plug-in coherence window field, specify the duration (in seconds, minutes, hours, or days) during which the contents of the data service are assumed to be fresh after the cache service has last inspected the back-end source object for updates. The default is 5 minutes.
- Select the user to execute the data service as: the user calling the service or a specific user. If you choose to execute the data service as a specific user, specify the qualified user name of a grid user who has permission to execute this database service. Use the following format:

`<user-name>@<authservice>.<authservicetype>.<domain>`

For example:

```
wilma@DefaultAuthService.Grid.Bedrock
```

- Step 4** Click **Next**. The system saves the new data expiration value and displays the View Data Services screen.

Viewing data services

To display a list of the data services in a cache service, do the following:

- Step 1** Navigate to the View Cache Services screen:

Home > Service management > View cache services

View Cache Services						
You are viewing the cache services in the local grid domain.						
Name	View/Edit	File Contents	DBOP Contents	Data Service Contents	Attributes	Security
MAGADIEUXP.sybase.com	View/Edit	File contents	DBOP contents	Data service contents	Attributes	Security
<input type="button" value="Done"/>						

The system displays a list of the cache services in the current domain.

- Step 2** Click the **Data Service Contents** link beside the name of the cache service whose data services you want to view. The View Data Service Cache screen appears, listing the data services in the cache service. The example below shows the four possible combinations of caching and invalidation conditions:

- **CherylDomain.MyDataService1:** This data service is scheduled to be executed and its results have been cached. You can invalidate the current cached results or unschedule the cache schedule.
- **CherylDomain.MyDataService2:** The data service result is cached on demand. You can invalidate the cached result.
- **CherylDomain.MyDataService3:** The cache service contains an entry for the data service and its cached metadata, but the cache service contains no cached results for the data service. This situation occurs if the data service's result is cached and then later invalidated.

- **CherylDomain.MyDataService4:** The cache service contains an entry for the data service and its metadata, but the data service results will not be cached.

View Data Service Cache

You are viewing the data services in the cache service on **blachmanxp.sybase.com**. To manage the caching of an entire data service, place a check mark beside the data service and click one of the following buttons:

- **Evict All Checked** untags, un schedules, and purges all cached result sets and metadata for the specified data services.
- **Invalidate All Checked** invalidates all of the cached result sets and metadata for the specified data services.

You can manage individual data service result sets by clicking the embedded **Invalidate** or **Unschedule** links to invalidate a result set or un schedule a particular cache schedule.

Data Service	Parameters	Status
<input type="checkbox"/> CherylDomain.MyDataService1	No Parameters [Invalidate]	Scheduled: Every 1 days, next execution Fri Jun 3 3:06:46 PM EDT 2005 [Unschedule]
<input type="checkbox"/> CherylDomain.MyDataService2	No Parameters [Invalidate]	Cached On-demand
<input type="checkbox"/> CherylDomain.MyDataService3	Not Applicable	Cached On-Demand (Metadata Only)
<input type="checkbox"/> CherylDomain.MyDataService4	Not Applicable	Data Caching Not Permitted (Metadata Cached)

Check All
Clear All

Evict All Checked
Invalidate All Checked
Done

Unscheduler/evicting data services

When you unschedule or evict a data service, the data service is purged from the cache service and unscheduled if it's pinned.

To unschedule or evict a scheduled data service, do the following:

Step 1 Navigate to the View Cache Services screen:

Home > Service management > View cache services

View Cache Services

You are viewing the cache services in the local grid domain.

Name	View/Edit	File Contents	DBOP Contents	Data Service Contents	Attributes	Security
MAGADIEUXP.sybase.com	View/Edit	File contents	DBOP contents	Data service contents	Attributes	Security

The system displays a list of the cache services in the current domain.

Step 2 Click the **Data Service Contents** link beside the name of the cache service that you want to unschedule or evict. The View Database Cache screen appears.

View Data Service Cache

You are viewing the data services in the cache service on **blachmanxp.sybase.com**. To manage the caching of an entire data service, place a check mark beside the data service and click one of the following buttons:

- **Evict All Checked** untags, unschedules, and purges all cached result sets and metadata for the specified data services.
- **Invalidate All Checked** invalidates all of the cached result sets and metadata for the specified data services.

You can manage individual data service result sets by clicking the embedded **Invalidate** or **Unschedule** links to invalidate a result set or unschedule a particular cache schedule.

Data Service	Parameters	Status
<input type="checkbox"/> CherylDomain.MyDataService	Employee: "FEIN" [Invalidate]	Cached On-demand
	Employee: "BLACHMAN" [Invalidate]	Cached On-demand

- Step 3** (Optional) Click boxes in the Query Name column to choose the database operations you want to unpin, uncache, or evict, then select one of the following:
- To unpin, uncache, and purge all cached results and metadata for the specified database operation, click **Evict All Checked**.
 - To invalidate all of the cached results data and metadata for the specified database operations, click **Invalidate All Checked**.
- Step 4** (Optional) To invalidate a specific cached result set, click the **Invalidate** link in the Parameters entry for that result set.
- Step 5** (Optional) To uncache a specific schedule, click the **Uncache** link beside the description of that schedule.

Configuring schedule exclusions

A schedule exclusion is a named time period that you define. When you apply a schedule exclusion to an entry in a schedule, the exclusion prevents the scheduled activity—such as a cache refresh, for example—from occurring during the time specified by the exclusion. A schedule exclusion can be applied to as many schedules as you like, and it can be applied to schedules for any scheduled activity, including rehashing Avaki shares, refreshing imported user accounts, and caching files, directories, and the results of database operations, virtual database operations, data services, and generated views.

For example, suppose you have a directory whose cache is scheduled to be refreshed once a day. This works well most of the time, but on the last day of the month, demand on the network (or the host computer) is very high and you want to reduce traffic. You can set up a schedule exclusion for the last day of every month and apply it to the directory's refresh schedule. If necessary, you can apply the same exclusion to other schedules to further reduce traffic. You can also configure each schedule so that the scheduled activity occurs before or after the schedule exclusion period, or not all.

This section includes these procedures:

- [“Setting up schedule exclusions” on page 167](#)
- [“Applying schedule exclusions to schedule entries” on page 169](#)

Setting up schedule exclusions

To define or edit a schedule exclusion, you must have write permission on the exclusion. In an out-of-the-box data grid, that means you must be a member of the DataProviders group.

Follow these steps to define a schedule exclusion.

Step 1 Navigate to the Add Schedule Exclusion screen:

Home > Service management > Create schedule exclusion.

Add Schedule Exclusion

Use this page to define a schedule exclusion. To do so, you must give it a name and specify one or more periods of time (either one-time or recurring) during which scheduled events would be blocked. The time zone that you specify for the blocked period can be relative to the grid server or relative to Greenwich Mean Time (GMT). At your option, you can also give this exclusion a description. When you're done, click **Submit** to add it to the set of defined exclusions; or click **Cancel** to discard it.

Note: Blocking scheduled events is a two-step process. The step you're doing here just defines an exclusion. The other step is to configure one or more event schedule(s) to respect the exclusion you're defining here. The exclusion you define here will affect only schedules that are explicitly configured to respect the exclusion.

Name:

Description (optional):

Select a time zone:

Type of schedule exclusion:

From to

Step 1 In the Name field, enter a name to identify this schedule exclusion. (Later, when you apply this exclusion to a schedule entry, you'll select this name from a list of exclusions.)

Step 2 (Optional) In the Description field, enter a description of this schedule exclusion.

Step 3 From the Time Zone pull-down, select the time zone in which you're specifying the exclusion period.

Step 4 Under Type of schedule exclusion, click a tab to specify how often you want the exclusion to recur:

- **Daily:** The exclusion blocks scheduled activities during the specified period every day. Use the pull-downs in the From and to fields to set the hour and minute at which the exclusion period begins and ends.
- **Weekly:** The exclusion blocks scheduled activities during the specified period on the specified days of every week. To set the duration of the exclusion, click the Entire day button or click the lower button, then use the pull-downs in the From and to fields to set the hour and minute at which the exclusion period begins and ends.

To set the days of the week, click the boxes for one or more days.

- **Monthly:** the exclusion blocks scheduled activities during the specified period on the specified days of every month. To set the duration of the exclusion, click the Entire day button or click the lower button, then use the pull-downs in the From and to fields to set the hour and minute at which the exclusion period begins and ends.

To set the days of the month, click the boxes for one or more days.

- **Yearly:** the exclusion blocks scheduled activities during the specified period on the specified days of every year. To set the duration of the exclusion, click the Entire day button or click the lower button, then use the pull-downs in the From and to fields to set the hour and minute at which the exclusion period begins and ends.

To set the days of the year, use the month and date pull-downs. Click Add day to add as many days as you need.

- **Custom:** the exclusion blocks scheduled activities during one or more periods that you define using the time, month, day, and year pull-downs. Click Add new range if you want to define additional time periods. Use the Custom tab to configure one-time exclusions.

Step 5 Click **Submit** to save your schedule exclusion.

The procedure that follows explains how to incorporate schedule exclusions into schedules.

Applying schedule exclusions to schedule entries

Before you can follow this procedure, at least one schedule exclusion must already be configured. See “[Setting up schedule exclusions](#),” above, for instructions.

To apply a schedule exclusion, you must have write permission on the object (the directory, for example) to which the schedule applies.

Follow these steps to apply a schedule exclusion.

- Step 1** Navigate to the view screen for the object to which the schedule applies. For example, for a database operation, navigate to the View Database Operations screen:

Home > Data provisioning > Manage database operations

View Database Operations

Grid domain:

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

The system displays a list of the database operations in the current domain.

- Step 2** Click the **Schedule** link beside the database operation you want to schedule. The Schedule Database Operation screen appears.

Schedule Database Operation

You have chosen to schedule the database operation **MyDBOperation** for periodic update in a cache service. You can choose to do this with any cache service in this domain. If you do it on the server local to the operation (shown with an asterisk * before its name), your scheduling will affect all requests using your indicated set of parameter values throughout the domain. If you schedule your updates on any other server, they will affect only requests made from that server. Next, specify the values of the input parameters for the operation (if there are any). Finally, click **Add Schedule** to go on to define the schedule on which your updates will occur, or **View Schedules** to review (and manage) any existing schedules for this combination of database operation, parameter values and server.

Cache service:

SQL statement: select * from emp where ename = ?

Parameter 1: VARCHAR

- Step 3** Select the cache service to use. If you want to reduce load on the underlying database, select the server that is local to the operation (the server with an asterisk * before its name) as the cache service. If you want to cache the results on a remote grid server (and thereby reduce network traffic when requests are made from that server), select the cache service for the related grid server.
- Step 4** Specify the values of the input parameters for the operation (if there are any).
- Step 5** Click **Add Schedule**.
- Step 6** The Add New Schedule screen appears.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

One time
 Periodic
 Calendared
 Advanced

Starting Now

12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur every: minutes ▼

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Schedule exclusions:

No schedule exclusions are currently defined in this domain.

- Step 7** Scroll down, if necessary, to expose the Schedule exclusions portion of the Add New Schedule screen.

Schedule exclusions:

Apply the following exclusions to this schedule: Midmonth
 Monday

When an exclusion conflicts with a scheduled occurrence, the system should:

cancel the occurrence
 reschedule the occurrence the conflicting exclusion, at an offset of

- Step 8** Select one or more schedule exclusions from the “Apply the following exclusions to this schedule” list. Use the up and down arrows to scroll if you don’t see the exclusion you want.

- Step 9** Click to specify what should happen when an exclusion prevents the cache from being refreshed on schedule:

- **Cancel the occurrence** causes the cache service to skip any scheduled refreshes that fall within the exclusion period.
- **Reschedule** causes the system to reschedule any scheduled refreshes that fall within the exclusion period.

- Step 10** If you chose Reschedule, use the pull-downs in the bottom line to specify:

- whether the system should reschedule the refresh for before or after the exclusion period; and
- how long before or after the exclusion period the system should try to reschedule the refresh.

For example, if you specify that the cache service should refresh 2 hours before the exclusion period begins, the system first tries to reschedule the refresh at the 2-hour point. If that time slot isn’t available, the system tries to reschedule the refresh for 4 hours before the exclusion period, then 6 hours, and so on at 2-hour intervals.

- Step 11** Click **Submit** to apply the specified exclusion and rescheduling policy to the refresh schedule.

Step 12 (Optional) To view details about schedule exclusions, do the following:

- Scroll down, if necessary, to the About Schedule Exclusions portion of the Add New Schedule screen.

About Schedule Exclusions

A schedule exclusion is a named set of time periods. When an exclusion is applied to an event schedule, it prevents the event from occurring during the designated periods. To create a new schedule exclusion, click **Add Exclusion**, or click **View Exclusions** to see the list of exclusions currently defined in this domain. *Note: Either choice will nullify any changes you've made to the current schedule.*

- Click **View Exclusions** to see the list of exclusions currently defined in this domain. The View Schedule Exclusions screen appears.

View Schedule Exclusions

You are viewing the schedule exclusions that have been defined in this grid domain.

Select	Name	Summary	View/Edit
<input type="checkbox"/>	Midmonth	Monthly exclusion on 15 from 12:00:00 AM to 11:59:00 PM	View/Edit
<input type="checkbox"/>	Monday	Weekly exclusion for Mon from 12:00:00 AM to 11:59:00 PM	View/Edit

- Optional. To view details about a particular schedule exclusion, click the **View/Edit** link beside the schedule exclusion whose details you want to view. The View/Edit Schedule Exclusion appears, showing details about the exclusion.

View/Edit Schedule Exclusion

This page allows you to change the description and the periods of time associated with this schedule exclusion. Click **Submit** for your changes to take effect, or **Cancel** to leave the exclusion unaltered.

Note: Blocking scheduled events is a two-step process. The step you're doing here just defines an exclusion. The other step is to configure one or more event schedule(s) to respect the exclusion you're defining here. The exclusion you define here will affect only schedules that are explicitly configured to respect the exclusion.

Name: Midmonth

Description (optional):

Select a time zone:

Type of schedule exclusion:

Entire day
 From : to :

1	2	3	4	5	6	7	8	9	10		
11	12	13	14	15	16	17	18	19	20		
21	22	23	24	25	26	27	28	29	30	31	last

- Click **Cancel** to return to the View Schedule Exclusions screen.

Setting up data service plug-ins

This chapter explains how to create and deploy data service plug-ins. It covers the following topics:

- [“Overview of data service plug-ins,”](#) below
- [“Java, JavaScript, or XSLT”](#) on page 176
- [“Input and output”](#) on page 177
- [“Plug-in files”](#) on page 180
- [“Deployment of plug-ins”](#) on page 180
- [“Creating XSLT plug-ins”](#) on page 181
- [“Creating Java plug-ins with the Plug-in Wizard”](#) on page 183
- [“Creating JavaScript plug-ins”](#) on page 200

Overview of data service plug-ins

An Avaki *data service* is a mechanism that can combine file or relational data from one or more sources. It transforms and/or merges the data and produces output that you can make available to an application, another data service, or an Avaki view generator.

The heart of a data service is a logic module called a *data service plug-in*. The logic can be written in Java or XSLT. Data service plug-ins are modular—you can make one plug-in serve several purposes by including it in different data services.

For example, suppose Fred has written a plug-in that accepts input from any RSS (Rich Site Summary or Really Simple Syndication) newsfeed and produces output in an HTML format of his own design. Fred can create a series of data services, each using the same RSS plug-in but specifying different inputs—one data service for the Washington Post, one for the BBC, and one combining material from several blogging sites. If Fred has other plug-ins, he might also create data services to re-use them. Using one plug-in, many data services can perform the same transformation on data from many input sources.

To set up an Avaki data service, you must complete these tasks:

1. Create a data service plug-in as described in this chapter.
2. Deploy the plug-in by sharing it into the Avaki data catalog.
3. Configure the data service in the web UI. For an overview of data services and instructions on setting them up, see [Chapter 2, “Basic data integration”](#).

Java, JavaScript, or XSLT

A plug-in can be implemented as a Java class, as JavaScript, or as an XSLT stylesheet:

- If your plug-in will be an XSLT stylesheet, you must write the stylesheet and share it into the data catalog. More information is available in the section [“Creating XSLT plug-ins” on page 181](#).
- Use Avaki’s Plug-in Wizard to create Java plug-ins. The Plug-in Wizard can help you create plug-ins with inputs and outputs in any format. However, the Plug-in Wizard produces only skeleton Java code; you must write the code that performs

the plug-in's key functions. Instructions on creating Java plug-ins appear in [“Creating Java plug-ins with the Plug-in Wizard” on page 183](#).

- A JavaScript plug-in consists of a single JavaScript file shared into the Avaki data catalog. Instructions on creating JavaScript plug-ins appear in [“Creating JavaScript plug-ins” on page 200](#).

Input and output

A data service plug-in, written either in Java or in XSLT, defines the data service's requirements for input sources, runtime parameters, and output stream, all of which are discussed in this section. A plug-in can have:

- Zero or more input sources
- Zero or more runtime parameters
- Zero or one output streams

Input sources

Each input source can consist of XML, rowset data (ResultSet type), or raw binary data (ByteArray type). (Avaki rowset format is an internal format for relational results. Data services and other Avaki services accept rowsets as input; data services and database operations produce rowset output.)

An input source can be a grid file or the result of running a database operation, or another data service. (Note, however, that you don't need to know the data source when you write a plug-in; the source is specified in the data service.)

Note If any of the inputs is a rowset—that is, it comes from a database operation or a data service that produces rowset output—and if the plug-in requires XML input, Avaki automatically generates an XML representation of the rowset data as it feeds it into the plug-in. (The XML is generated on the fly, in the JVM as it is consumed by the plug-in, so XML is not flowing across servers.) The schema for the generated XML is that which Avaki generates for database operations; a schema description appears in [Appendix D, “Avaki rowset XML”](#).

If your plug-in is a stylesheet, you will specify exactly two input sources when you set up a data service for it:

- **The primary input**
The primary input source for an XSLT plug-in is the document that the style sheet is applied to. The primary input is defined only in the data service; it does not appear in the stylesheet. The primary input is a single input stream for which you must specify a name and a grid source.
- **The secondary input**
The secondary input is configured in the data service as a list of all input sources other than the primary input; each item in the list is an input source for which you must specify a name and a grid source. (There's nothing special about secondary inputs; they just need to be specified separately from the primary input.) If your stylesheet has only one input source, you can leave the secondary input list empty. To set up secondary inputs in your stylesheet, you use the `XPATH document ()` function with the `avaki://` syntax for referencing grid objects, as described in [“Specifying secondary input sources” on page 181](#).

If your plug-in is a Java class, you can specify any number of input sources—there is no notion of primary or secondary. Each input source is available to your plug-in's `run ()` method as an instance of an `InputStream` (see [“InputStream interface” on page 186](#)).

Parameters

Runtime parameters can be divided into three categories:

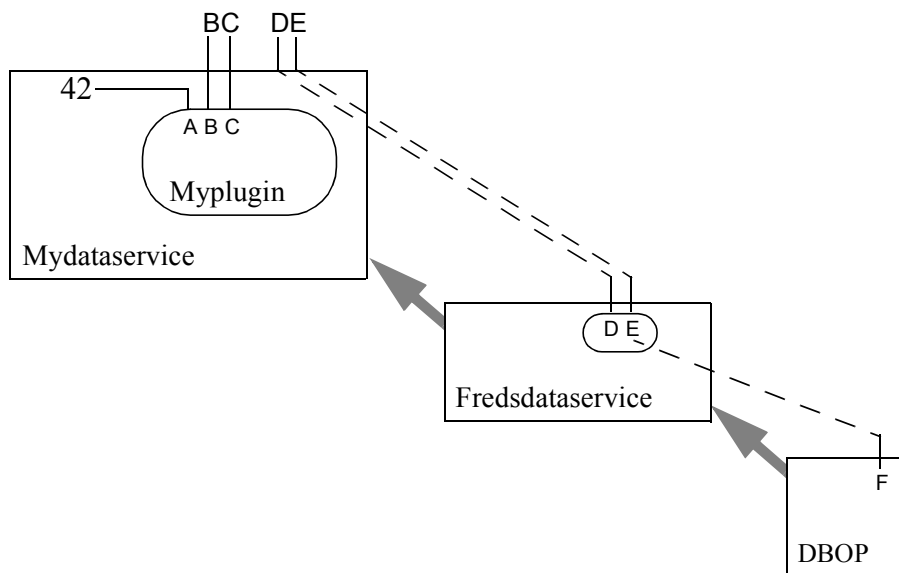
- **Plug-in parameters**
Parameters required by the plug-in. When you set up a data service for this plug-in, you'll bind each plug-in parameter either to a constant value (so that no value needs to be provided at run-time) or to a data service parameter.
- **Input source parameters**
Parameters required by input sources. Suppose you've constructed a plug-in whose input sources include a data service (or a database operation), and that data service has runtime parameters of its own. Those parameters are input source parameters. In data services that use your plug-in, you'll bind each input source parameter either to a constant value or to a data service parameter.
- **Data service parameters**
Parameters required by the data service. Values for data service parameters must be provided at runtime. Data service parameters are typically bound to plug-in parameters or input source parameters.

In the figure below, Myplugin has three plug-in parameters, A, B, and C. In Mydataservice, A is bound to a constant value, 42, so it does not require input at run-time. Mydataservice has an input source called Fredsdataservice, and Fredsdataservice has parameters of its own, D and E. Parameters B, C, D, and E are bound to corresponding data service parameters in Mydataservice.

Parameter A is a plug-in parameter, but not a data service parameter. B and C are both plug-in parameters and data service parameters. With respect to Myplugin, D and E are both input source parameters and data service parameters.

Fredsdataservice has an input source called DBOP that calls for a parameter F. At run-time, the values for parameters D and E are passed from Mydataservice to Fredsdataservice; the value for parameter E is passed on to the DBOP, whose parameter F uses it.

Parameters A, B, C, D, and E must be defined in Mydataservice. Mydataservice makes no reference to parameter F, but parameter E's value is passed to F via Fredsdataservice.



To set up Myplugin and Mydataservice, you only need to know about the parameters required by Fredsdataservice; Fredsdataservice is responsible for collecting any parameters required by its input sources (in this case, DBOP).

Output stream

The plug-in's output stream, if it has one, is bound to the data service's output stream when the data service is configured. It may consist of XML, rowset data (ResultSet type), or raw binary data (ByteArray type).

Plug-in files

JAR files and manifest files for Java plug-ins

A data service plug-in written in Java is deployed in a JAR (Java archive) file. The JAR file must contain:

- The Java code that implements the plug-in
- A manifest file that describes the components of the plug-in, such as required parameters, input sources, and output streams

The Plug-in Wizard creates the manifest file for you. It also creates build scripts for compiling the code and setting up the JAR.

JavaScript file for JavaScript plug-ins

A JavaScript plug-in consists of a JavaScript file.

XSL file for XSLT plug-ins

An XSLT plug-in consists of an XSL stylesheet.

Deployment of plug-ins

Once you've created the JAR file or the XSLT stylesheet for your plug-in, you'll create an Avaki share that imports the plug-in into the grid data catalog, or copy the plug-in into an existing Avaki share. Your plug-in is then ready to be used in a data service. Configuring data services in the web UI is covered in [Chapter 2, "Basic data integration"](#).

Creating XSLT plug-ins

To set up a data service plug-in that performs XSLT transformations, you must:

1. Write a stylesheet in XSLT. This section explains how to set up parameters and secondary input sources in the stylesheet so that they can be configured in a data service later.
2. Share the stylesheet into the grid data catalog. (Instructions on creating Avaki shares can be found in the *Sybase Avaki EII Administration Guide*.)

Avaki provides two XSLT engines: Saxon and Xalan. Your stylesheet should be compatible with one of these engines.

Specifying parameters

You can use `xsl:param` elements in your stylesheet to refer to plug-in parameters. When you create a data service for this plug-in, you'll be asked to do one of the following for each parameter:

- Specify a constant value for the parameter, or
- Bind the plug-in parameter to a data service parameter, for which a value must be provided at run time.

Specifying secondary input sources

In your stylesheet, you can use the `document()` function to refer to secondary input sources. When you create a data service for this plug-in, you'll be asked to bind these named secondary inputs to grid sources.

The `document()` function fetches and parses XML data from a specified URI. Avaki allows you to provide the URI in two special formats for sources in the data catalog (in addition to the usual syntax that allows you to specify local files, HTTP URLs, and so forth):

- `avaki://<full-grid-path>`
For example, `avaki:///Shares/myfiles/paris.xml`. Use this format to specify files only.
- `avaki://<name>`
For example, `avaki://EmpData`. The `<name>` is an arbitrary name you assign to this input source. When you create a data service that uses this stylesheet as its plug-in, you'll be asked to bind this input source name to a file, a database operation, or another data service.

A sample XSLT plug-in

The merge.xsl stylesheet describes how to merge a set of photos in .jpg files and text from a series of XML files to produce a single HTML file. (For more information on how this merge works, see <avaki-install-dir>/examples/merge/README.txt.) The merge.xsl file specifies two items of interest, a parameter and an input source:

- An `xsl:param` tag defines a parameter called `title`.
- An `xsl:apply-templates` tag uses a `document()` function to open and parse a file. (The file name is `.`, the value of the current XML element—in this case, `ph:entry`.)

The contents of merge.xsl are as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:ph="http://ananas.org/2003/tips/photo">
  <xsl:output method="html" encoding="utf-8" />

  <xsl:param name="title"/> ←————

  <xsl:template match="ph:index">
    <html>
      <head>
        <title>
          <xsl:value-of select="$title" />
        </title>
      </head>

      <xsl:apply-templates />
    </html>
  </xsl:template>

  <xsl:template match="ph:index/ph:title">
    <h1>
      <xsl:apply-templates />
    </h1>
  </xsl:template>

  <xsl:template match="ph:entry">
    

    <xsl:apply-templates select="document(concat('avaki://',.))" /> ←————

    <br clear="right" />
  </xsl:template>

  <xsl:template match="ph:photo/ph:title">
```

```

    <h2>
      <xsl:apply-templates />
    </h2>
</xsl:template>

<xsl:template match="ph:location">
  <h3>in
  <xsl:apply-templates />
</h3>
</xsl:template>

<xsl:template match="ph:date">
  <p>Date:
  <xsl:apply-templates />
</p>
</xsl:template>

<xsl:template match="ph:description">
  <p>
    <xsl:apply-templates />
  </p>
</xsl:template>
</xsl:stylesheet>

```

Creating Java plug-ins with the Plug-in Wizard

You can use the Avaki Plug-in Wizard to generate skeleton Java code and build.xml and manifest files for a data service plug-in. The skeleton Java code of the plug-in contains a comment at the point where you must insert your own Java code that performs the merge or transformation.

The Plug-in Wizard runs from the command line in the local directory where Avaki software is installed.

In this section:

- [“Prerequisites,”](#) below
- [“Plug-in Wizard procedure”](#) on page 184
- [“Writing the Java code”](#) on page 185

Prerequisites

To prepare to create plug-ins using Avaki’s Plug-in Wizard, you need the following:

- Avaki 5.0 or later installed on at least one machine, with at least a grid domain controller running
- A Java compiler such as Sun JDK 1.4
- Apache Ant 1.5.3.1 or later (required to compile the Java source code for your plug-ins). Ant is free; you can download it from <http://ant.apache.org>.

Plug-in Wizard procedure

Follow these steps to create a plug-in using the Plug-in Wizard.

- Step 1** Ensure that the software listed under “Prerequisites,” above, is in place and ready to use.
- Step 2** To invoke the Plug-in Wizard, execute the **avaki plugin --generate** command. (For details on this command and its options, see the *Sybase Avaki EII Command Reference*.) For example, you might enter:

```
C:\AvakiDataGrid70> avaki plugin --generate
--plugin-name="MyPlugin" --impl-class=MyPluginClass
--method-name=run --parameter="name=p1;type=INTEGER"
--input="name=input1;type=ResultSet"
--input="name=input2;type=ResultSet"
--target-dir="c:\bedrock\wilma\plugins"
--template-dir="c:\bedrock\wilma\Javaki\templates\plugin"
--output="name=output1;type=ResultSet"
```

The Plug-in Wizard produces skeleton Java source code for the plug-in in a file called MyPlugin.java; a build.xml file; and a manifest file.

Note When you create a data service for this plug-in, you’ll specify the sources of the input streams named in the **avaki plugin** command (input1 and input2 in the example above). Input1 and input2 above consist of ResultSet data, so their sources will probably be Avaki database operations or data services. You can also configure the data service to supply static values for any parameters, or allow the values to be supplied at run time.

- Step 3** In the skeleton code, find the comment that says “put your logic here.” At that point, insert your own Java code that performs the desired merge or transformation. See “Writing the Java code” on page 185 for information and examples.

- Step 4** Use Ant to compile the Java source code with the build.xml file. (Configure the build.xml file to point to your ANT_HOME and run it.)

The result will be a JAR file that contains the compiled Java code and the manifest file. The JAR file is the plug-in.

- Step 5** Share the JAR file into the data catalog so that it can be incorporated into a data service. You can either copy the JAR into an existing Avaki share or create a new Avaki share. For instructions on creating Avaki shares, see the *Sybase Avaki EII Administration Guide*. For instructions on setting up a data service that uses your new plug-in, see [Chapter 2, “Basic data integration”](#).
- Step 6** To improve performance for data services that use this plug-in, pin the JAR file for caching. (Executing a data service just once can require multiple reads of the JAR file. Caching the JAR file can reduce the read time considerably.) For instructions on pinning a file for caching, see [“Pinning files for scheduled caching” on page 108](#).

Writing the Java code

Avaki provides an API that data service plug-ins can use to access their inputs, manipulate their output, and access various services. A second API is provided for building data services that group database operations to perform distributed transactions. We assume that users of the data service and transaction APIs have working knowledge of Java, including, in particular, knowledge of Sun’s JDBC and I/O APIs.

This section describes the interfaces and classes in the Avaki APIs that are most commonly used in data services, then provides examples showing how to use them in your Java code. For further details on the Avaki APIs, see the Javadoc, which you can view on any machine that has network access to an Avaki installation. To view the Javadoc, point your browser to <Avaki-install-dir>/docs/api/index.html. For example, you might enter:

```
C:\AvakiDataGrid70\docs\api\index.html
```

On the overview page, click **com.avaki.core.services.dataservice.api** for the general data service API, or **com.avaki.transaction** for the distributed transaction API.

The remainder of this section consists of reference information on the Avaki API interfaces and classes that are most commonly used in data service plug-ins, and on related topics including logging and manifest files:

- [“The Avaki Data Service API,”](#) below
- [“The Avaki Transaction API” on page 189](#)

- [“Code samples for Java data service plug-ins” on page 191](#)
- [“Logging” on page 196](#)
- [“Manifest files and build.xml files” on page 197](#)

The Avaki Data Service API

Use the data service API for all Java plug-ins, including those that perform distributed transactions.

Every Java plug-in must implement the `Plugin` interface and include a manifest file.

This section includes the following subsections:

- [“Plugin interface,” below](#)
- [“InputSource interface” on page 186](#)
- [“ParameterSpec interface” on page 187](#)
- [“StreamingRowSet interface” on page 187](#)
- [“RowSetFactory class” on page 188](#)

Note There is no need to close the data service output stream in the code you add to your plug-in; the data service itself will close the output stream. However, be sure to close or flush any other streams in your code, such as stream decorators that buffer output. (Use `closeRowSet()` for the Avaki `StreamingRowSet` interface, `endDocument()` for `org.apache.xml.serialize.XMLSerializer`, or `flush()` for `BufferedWriter`.)

Plugin interface. The `Plugin` interface provides the custom data transformation logic for the data service. `Plugin` has one method, `run`. For more information on `Plugin`, see the Javadoc for the data service API.

InputSource interface. The `InputSource` interface of the data service API provides the abstraction for invoking and accessing the results of database operations and data services, as well as accessing file content, from within a data service plug-in. [“Example: Rowset input, rowset output” on page 192](#) uses `InputSource` to get input from a database operation. [“Example: Merge a DBOP result and a CSV file to produce XML output” on page 194](#) uses `InputSource` to access both the results of a database operation and the contents of a file.

The table that follows lists the methods for `InputSource`. For more information on `InputSource`, see the Javadoc for the data service API.

<code>void</code>	<code>execute()</code> Executes the underlying <code>InputSource</code> .
<code>StreamType</code>	<code>getDataType()</code> Returns the type of the data this stream will provide.
<code>Object</code>	<code>getInputParameter(ParameterSpec param)</code> Returns an input parameter of this input source.
<code>InputStream</code>	<code>getInputStream()</code> Returns a stream for reading data from.
<code>ResultSet</code>	<code>getResultSet()</code> Returns the data as a result set.
<code>Set</code>	<code>getSupportedInputParams()</code> Returns a set of <code>ParameterSpec</code> s representing the input parameters this input source expects.
<code>Object</code>	<code>getTarget()</code> Returns the target of this input source, either as a <code>String</code> (if it's set directly) or as a <code>ParameterSpec</code> (if it's a reference to a data service parameter).
<code>String</code>	<code>resolveTarget()</code> Returns the path to the target of this input source, resolving a parameter if it's set to one.
<code>void</code>	<code>setInputParameter(ParameterSpec param, Object value)</code> Sets an input parameter on this input source.

ParameterSpec interface. When parameters are specified in the **avaki plugin** command, the Plug-in Wizard generates skeleton code using the `ParameterSpec` interface, so there's typically no need to add it to the plug-in yourself. The `ParameterSpec` interface of the data service API provides the abstraction for manipulating parameters to your plug-in and to its input sources. You can retrieve `ParameterSpec`s for parameters passed to your plug-in as well as for parameters that have been configured for the various input sources. You can also set and modify parameters to input sources before invoking them using the `ParameterSpec` abstraction. For more information on `ParameterSpec`, see the Javadoc for the data service API.

StreamingRowSet interface. `StreamingRowSet`, which implements the `javax.sql.RowSet` interface, lets you construct an Avaki rowset for the output of your data service. `StreamingRowSet` writes directly to an output stream; it does not allow

iteration over the set. To populate the rowset, you can use methods on `javax.sql.RowSet` (see Sun’s Javadoc at <http://java.sun.com/j2se/>) as well as Avaki’s extensions, below. Note that when you’re finished writing to the rowset, you must use the `closeRowSet()` method. “[Example: Rowset input, rowset output](#)” on [page 192](#) uses `StreamingRowSet`.

The table that follows lists the methods for `StreamingRowSet`. For more information on `StreamingRowSet`, see the Javadoc for the data service API.

<code>void</code>	<code>addRow()</code>	Adds a new row to the end of this rowset.
<code>void</code>	<code>closeRowSet()</code>	(Required.) Closes the rowset and flushes all data to the stream.

RowSetFactory class. The `RowSetFactory` class of the data service API creates Avaki rowsets. Use this class to implement a plug-in whose output is in rowset format. Rowset output is useful as input to other data services. You can use `RowSetFactory` to create both in-memory rowsets (which allow you to iterate over the data) and streaming rowsets (for which iteration is not possible).

When you create a rowset, you must pass in an array of integers from the class `java.sql.Types`. The integers correspond to the types of the columns in the rowset. After creating the rowset, you might want to add more detailed rowset metadata. You can do so by calling the method `getMetaData`, casting the results to the `javax.sql.rowsetMetaData` method, then using setter methods to add the additional metadata to the rowset.

“[Example: Rowset input, rowset output](#)” on [page 192](#) uses `RowSetFactory` to create streaming rowsets.

The table that follows lists the methods for `RowSetFactory`. For more information on `RowSetFactory`, see the Javadoc for the data service API.

<code>static</code>	<code>createMemoryRowSet(int [] sqlTypes)</code>	
<code>MemoryRowSet</code>		Generates an in-memory rowset that can be streamed at any time.
<code>static</code>	<code>createStreamingRowSet(int [] sqlTypes)</code>	
<code>Streaming-RowSet</code>		Generates a streaming rowset backed by a temporary file.
<code>static</code>	<code>createStreamingRowSet(int [] sqlTypes, OutputStream os)</code>	
<code>Streaming-RowSet</code>		Generates a streaming rowset backed by a user-specified stream.

The Avaki Transaction API

Use the transaction API along with the data service API for Java plug-ins that execute distributed transactions.

Here's how to execute a distributed transaction using the transaction API:

- Step 1** Create an instance of the `XAWorkUnit` interface that describes the work to be done. You can write your own, or use one of the implementations provided in the transaction API: `DbopGroupWorkUnit` or `DbopPipeWorkUnit`.
- Step 2** Create an instance of the `XAWorkHandler` class (passing it the `XAWorkUnit` from [Step 1](#)) and call its `execute` method.
- Step 3** If you need to look at transaction by-products (such as `ResultSet`s from database operations), query your `XAWorkUnit` (using its custom accessor methods) for any data that was stored during the transaction execution. Process this data in whatever way you like; for example, you might write the data to the plug-in's output stream.
- Step 4** If your `XAWorkUnit` supplies a method for cleaning up its stored results, call this `cleanup` method.

XAWorkUnit interface. `XAWorkUnit` describes the work to be done in the distributed transaction, including the data sources that will participate in the transaction and the logic to be performed on the data sources.

The table that follows lists the methods for `XAWorkUnit`. For more information on `XAWorkUnit`, see the Javadoc for the transaction API.

<code>DistributedAction</code>	<code>getAction()</code> Returns the logic to be performed inside the distributed transaction.
<code>XADataSource</code>	<code>getDataSources()</code> Returns the data sources that will participate in the distributed transaction.

XAWorkHandler class. `XAWorkHandler` executes a distributed transaction described by `XAWorkUnit`. `XAWorkHandler` has one method, `execute()`.

DbopGroupWorkUnit class. `DbopGroupWorkUnit` executes a set of database operations in one distributed transaction.

The table that follows lists the methods for `DbopGroupWorkUnit`. For more information on `DbopGroupWorkUnit`, see the Javadoc for the transaction API.

<code>void</code>	<code>cleanup()</code> Cleans up any results that were stored during execution.
<code>DistributedAction</code>	<code>getAction()</code> Returns the <code>DistributedAction</code> in this work unit.
<code>XADataSource []</code>	<code>getDataSources()</code> Returns an array of <code>XADataSources</code> corresponding to the database operations in this work unit.
<code>com.avaki.core.services.database.JDBCExecutionOutput []</code>	<code>getExecutionOutputs()</code> Returns an array of <code>JDBCExecutionOutput</code> objects containing the results of executing the database operations in this <code>DbopGroupWorkUnit</code> .

DbopPipeWorkUnit class. `DbopPipeWorkUnit` executes one source database operation and passes the results to one or more target database operations, which it also executes, all within one distributed transaction. [“Example: Distributed transaction” on page 191](#) uses `DbopPipeWorkUnit`.

The table that follows lists the methods for `DbopPipeWorkUnit`. For more information on `DbopPipeWorkUnit`, see the Javadoc for the transaction API.

<code>void</code>	<code>cleanup()</code> Cleans up any results that were stored during execution.
<code>DistributedAction</code>	<code>getAction()</code> Returns the logic to be performed inside the distributed transaction.
<code>XADataSource []</code>	<code>getDataSources()</code> Returns the data sources that will participate in the distributed transaction.
<code>com.avaki.core.services.database.JDBCExecutionOutput</code>	<code>getSourceExecutionOutput()</code> Returns a <code>JDBCExecutionOutput</code> object containing the result of executing the source database operation in this <code>DbopPipeWorkUnit</code> .

```

com.avaki.core.services.data-   getTargetExecutionOutputs()
base.JDBCExecutionOutput []    Returns an array of JDBCExecutionOutput objects
                               containing the results of executing the target data-
                               base operations in this DbopPipeWorkUnit.

```

Code samples for Java data service plug-ins

In this section:

- [“Example: Distributed transaction,”](#) below
- [“Example: Rowset input, rowset output”](#) on page 192
- [“Example: Merge a DBOP result and a CSV file to produce XML output”](#) on page 194

Example: Distributed transaction. The example that follows shows the run method of a data service plug-in that expects three DbopInputSources. The plug-in constructs and executes a DbopPipeWorkUnit that copies data from the first database operation into the second and third. Note that when this plug-in is deployed, the input sources must be mapped to appropriate database operations—not every database operation will be compatible with the operation being performed. All the database operations must use XA-enabled database connectors and each of the back-end databases must be reachable from the grid server where the data service runs. The second and third database operations must be update DBOPs (that is, they must write to the database) and must expect parameters that correspond exactly (in number and type) to the output columns of the first database operation.

Here’s the code to be added to the `plugin()` method of the skeleton plug-in:

```

public void runThreeDBOPpipe(InputSource dbop1, InputSource dbop2, InputSource
dbop3, OutputStream XMLOutput) throws Exception {

    DbopPipeWorkUnit work = new DbopPipeWorkUnit((DbopInputSource)dbop1, new
DbopInputSource [] {(DbopInputSource)dbop2, (DbopInputSource)dbop3});

    // If you want to use your own TransactionManager implementation,
    // pass it in here when creating the XAWorkHandler
    XAWorkHandler handler = new XAWorkHandler(work);

    try {
        handler.execute();

        JDBCExecutionOutput sourceOutput = work.getSourceExecutionOutput();
        JDBCExecutionOutput [] targetOutputArray =
work.getTargetExecutionOutputs();

```

```

// Add header to output
String obj = "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>";
byte[] header = obj.getBytes();
XMLOutput.write(header, 0, header.length);

// Add root tag to output
obj = "<allResultSets>";
byte[] rootTag = obj.getBytes();
XMLOutput.write(rootTag, 0, rootTag.length);

// Add all result set data to output
sourceOutput.toXML(XMLOutput);
for (int i = 0; i < targetOutputArray.length; i++)
{
    targetOutputArray[i].toXML(XMLOutput);
}

// Close root tag
obj = "</allResultSets>";
byte[] endRootTag = obj.getBytes();
XMLOutput.write(endRootTag, 0, endRootTag.length);

} finally {
    work.cleanup();
}
}

```

Example: Rowset input, rowset output. The example that follows shows how to do a simple rowset (ResultSet) to rowset conversion. It takes input from a database operation and produces another rowset. This example uses the RowSetFactory class and the StreamingRowSet and InputSource interfaces from Avaki's data services API. Notice that the code portion of the example begins with imports required to compile.

Here's the **avaki plugin** command that generates the skeleton of the plug-in:

```

C:\AvakiDataGrid70> avaki plugin --generate
--plugin-name=ResultSetCopy
--package=com.avaki.plugin.sample
--input="name=resultSetInput;type=ResultSet"
--output="name=resultSetOutput;type=ResultSet"
--target-dir="c:\plugins\plugin1"

```

Here's the code to be added to the plugin() method of the skeleton plug-in:

```

// Added imports
import java.sql.ResultSet;
import java.sql.Types;

```



```

import javax.sql.RowSetMetaData;
import java.sql.ResultSetMetaData;
import com.avaki.core.services.dataservice.api.
    RowSetFactory;
import com.avaki.core.services.dataservice.api.
    StreamingRowSet;

/**
 * This method implements the user-defined plug-in
 * logic.
 *
 * This logic does a simple ResultSet conversion
 * performing a numerical operation
 */

public void plugin(InputSource resultSetInput,
    OutputStream resultSetOutput) throws Exception {

    // Grab the input result set. The hypothetical format
    // for this rowset is (VARCHAR(10), DOUBLE)
    ResultSet rs = resultSetInput.getResultSet();

    // Capture the input metadata
    ResultSetMetaData inputMetaData = rs.getMetaData();

    // Create a streaming rowset for the output
    int[] types = new int[2];
    types[0] = Types.VARCHAR;
    types[1] = Types.DOUBLE;
    StreamingRowSet rowSet = RowSetFactory.
        createStreamingRowSet(types, resultSetOutput);
    // Add some more metadata information. Note the cast
    // here.
    RowSetMetaData outputMetaData = (RowSetMetaData)
        rowSet.getMetaData();
    for (int i = 1; i <= 2; i++) {
        outputMetaData.setPrecision(i, inputMetaData.
            getPrecision(i));
        outputMetaData.setScale(i, inputMetaData.
            getScale(i));
    }
    // Copy the rows doing a simple transformation
    while (rs.next()) {
        rowSet.addRow();
        rowSet.setString(1, rs.getString(1));
        rowSet.setDouble(2, rs.getDouble(2) * 365);
    }
    // Make sure to close the rowset
    rowSet.closeRowSet();
}

```

Example: Merge a DBOP result and a CSV file to produce XML output. The next example shows plug-in code that takes two inputs and merges them. The first input source is a result set; the second is a flat CSV file. (The result set might come from a database operation; the file might be in the data grid or a local file system—the sources are specified in the data service.) The inputs are merged into an XML output file. Two parameters, `col1name` and `col2name`, define the column names for the CSV input. This example uses the `InputSource` interface from Avaki's data services API.

The command to generate the skeleton of the plug-in looks like this:

```
C:\AvakiDataGrid70> avaki plugin --generate
--plugin-name=CSVMerge --package=com.avaki.plugins.samples
--parameter="name=col1name;type=VARCHAR"
--parameter="name=col2name;type=VARCHAR"
--input="name=DBOPInput;type=ResultSet"
--input="name=CSVInput;type=XSL"
--output="name=XMLOutput;type=XML"
--target-dir="f:\plugins\CSVMerge"
```

Here's the code to be added to the `plugin()` method of the skeleton plug-in:

```
// Added imports
import java.sql.*;
import java.io.*;
import java.util.StringTokenizer;

// Used to build XML
import org.apache.xml.serialize.XMLSerializer;
import org.apache.xml.serialize.OutputFormat;
import org.xml.sax.helpers.AttributesImpl;

/**
 * This plug-in takes two inputs. First is a result set
 * that comes from a DBOP. Second is a flat file CSV
 * from a grid file. The inputs are merged into an XML
 * output. There are two parameters that define the
 * column names for the CSV input.
 */
public void plugin(java.lang.String col1name,
    java.lang.String col2name, InputSource DBOPInput,
    InputSource CSVInput, OutputStream XMLOutput) throws
    Exception {

    // Grab the input result set. The hypothetical format
    // for this rowset is (VARCHAR(10), DOUBLE)
    ResultSet rs = DBOPInput.getResultSet();

    // Capture the input metadata
    ResultSetMetaData resultSetMetaData =
        rs.getMetaData();
```

```

// Set up a reader for the input file
BufferedReader reader = new BufferedReader(new
    InputStreamReader(CSVInput.getInputStream()));

// Set up XML serializer for output
OutputFormat outputFormat = new OutputFormat("xml",
    "UTF-8", true);
OutputStreamWriter writer = new
    OutputStreamWriter(XMLOutput, "UTF-8");
XMLSerializer xmlSerializer = new
    XMLSerializer(writer, outputFormat);
AttributesImpl emptyAttributeList = new
    AttributesImpl();

// Create a root tag
xmlSerializer.startDocument();
xmlSerializer.startElement("", "", "rowset",
    emptyAttributeList);
// Iterate through the result set
while (rs.next()) {

    // Grab a row from the CSV
    String csvLine = reader.readLine();

    // If we have no more data then break
    if (csvLine == null) {
        break;
    }

    // Create a tag for the row
    xmlSerializer.startElement("", "", "row",
        emptyAttributeList);

    // Output the result set data
    for (int i = 1; i <= resultSetMetaData.
        getColumnCount(); i++) {

        // Get the column name to use as a tag name
        String colName = resultSetMetaData.
            getColumnName(i);
        xmlSerializer.startElement("", "", colName,
            emptyAttributeList);

        // Output the column data
        Object obj = rs.getObject(i);

        // Handle null columns
        if (obj == null) {
            obj = new String("null");
        }
        char[] colData = obj.toString().toCharArray();
        xmlSerializer.characters(colData, 0,

```

```

        colData.length);

        // Close the column data
        xmlSerializer.endElement("", "", colName);
    }

    // Output the CSV data
    int i = 0;
    StringTokenizer tokenizer = new StringTokenizer
        (csvLine, ",");
    while (tokenizer.hasMoreTokens())
    {
        i++;
        String colName = (i == 1) ? col1name : col2name;
        xmlSerializer.startElement("", "", colName,
            emptyAttributeList);

        char[] colData = tokenizer.nextToken().
            toCharArray();
        xmlSerializer.characters(colData, 0,
            colData.length);

        // Close the column data
        xmlSerializer.endElement("", "", colName);
    }

    // Create a tag for the row
    xmlSerializer.endElement("", "", "row");
}

// Close root tag
xmlSerializer.endElement("", "", "rowset");
xmlSerializer.endDocument();
}

```

Logging

Avaki uses log4j for logging. If a plug-in throws an exception, a log4j event is raised on the grid server where the data service is executing; the log properties file (log4j.xml) on that grid server determines how the message is logged. Under the default log4j configuration, a message is written to the grid server's log file—`<Avaki-install-dir>/jboss/server/grid-server/log/server.log`. However, you can configure a log4j appender to send you e-mail when errors related to your plug-in occur, or to write messages to some central place.

You can use log4j from within your plug-in to log messages in some category of your choosing, then configure log4j to do special things with messages in that category. The code might look something like this:

```
Logger myLogger = Logger.getLogger("plugins.MyPlugin");
myLogger.info ("Here's an info message");
myLogger.error ("Here's a more severe error");
```

You can then edit the log4j.xml file on the Avaki server where your data service runs to configure special rules for your category (plugins.MyPlugin) and for different severity levels. See the *Sybase Avaki EII Administration Guide* for information on log4j.xml files.

Note If you're using pooled execution services, your data service may run on any grid server in the pool, and it may run on a different server each time it runs. In this situation, configure log4j on all the grid servers in the pool to log messages from your plug-in in the same way—for example, by writing them to your home directory.

Manifest files and build.xml files

This section describes fields used in both the manifest file and the build.xml file for a data service Java plug-in.

Fields. The manifest file specifies the required parameters, input sources, and output streams for the data service. The fields that specify parameters, inputs, and output are described in the table below. (These fields work in both manifest files and in build.xml files, though in build.xml files the fields are embedded in XML.) Sample manifest and build.xml files follow the table.

Note Three sets of fields in the table end with numbers: the input fields, the output fields, and the parameter fields. Use an appropriate set of fields to describe each of your plug-in's input streams, each of its parameters, and its output stream. All the fields in a set that end with the same number describe the same object. For example, fields that begin with "input" and end with 0—inputname-0, inputtype-0, inputislist-0, and inputdesc-0—all describe the same input stream. The numbering sequence for each set of fields must start with 0.

Field	Value	Example
Name	(Required) The name of the section of the manifest file that defines the plug-in. The value of this field must be Avaki-plugin.	Name: Avaki-plugin
plugin-name	(Required) Supply a name for the data service plug-in. The default is myplugin.	plugin-name: myplugin
plugin-class	(Required) The name of the class that implements the plug-in.	plugin-class: com.avaki.AvakiMapping
plugin-desc	(Optional) A description of this plug-in.	plugin-desc: Departmental expense reports
inputname-<#>	A unique name for an input stream.	inputname-0: mf_ExReport.Main1
Note: The names of all fields relating to a given input stream must end with the same number.		
inputtype-<#>	The input stream type, which can be XML (for custom data in XML format), ResultSet (for database result rowsets), or ByteStream (for raw binary data).	inputtype-0: XML
inputislist-<#>	Indicate whether this input stream is a list or a singleton. The possible values are true or false. The default is false.	inputislist-0: false
inputdesc-<#>	(Optional) A description of the input stream.	inputdesc-0: Expense data from travel database
outputname-0	A unique name for the output stream.	outputname-0: ExpReport_Target.Main1
outputtype-0	The output stream type, which can be XML (for custom data in XML format), ResultSet (for database result rowsets), or ByteStream (for raw binary data).	outputtype-0: XML
outputislist-0	Indicate whether the output stream is a list or a singleton. The possible values are true or false. The default is false.	outputislist-0: false

Field	Value	Example
outputdesc-0	(Optional) A description of the output stream.	outputdesc-0: Not specified
paramname-<#>	A unique name for a plug-in parameter.	paramname-0: DeptName
<p>Note: All the fields relating to this parameter stream must end with the same number.</p>		
paramtype-<#>	The type of the plug-in parameter, which must be one of the following: ARRAY, BIGINT, BINARY, BIT, BOOLEAN, BLOB, CHAR, CLOB, DATE, DECIMAL, DOUBLEFLOAT, INTEGER, JAVA_OBJECT, LONGVARCHAR, NUMERIC, REAL, SMALLINT, TIME, TIMESTAMP, TINYINT, VARBINARY, VARCHAR	paramtype-0: VARCHAR
paramislist-<#>	Indicate whether this parameter's value is a list or a singleton. The possible values are true or false. The default is false.	paramislist-0: false
paramdesc-<#>	A description of this plug-in parameter.	paramdesc-0: The department running the expense report

Sample manifest file. Here is an example of a manifest file. Note that no blank lines are allowed within the section of the manifest file that defines the plug-in. (Blank lines within the plug-in section cause the manifest file to be parsed incorrectly.)

```
Manifest-version: 1.0
Created-By: 1.4.1_03-b02 (Sun Microsystems Inc.)

Ant-Version: Apache Ant 1.5.3
Name: Avaki-plugin
plugin-name: myplugin
plugin-desc: Departmental expense reports
plugin-class: com.avaki.AvakiMapping
inputname-0: mf_ExReport.Main1
inputtype-0: XML
inputislist-0: false
inputdesc-0: Expense data from travel database
outputname-0: ExpReport_Target.Main1
outputtype-0: XML
outputislist-0: false
outputdesc-0: Not specified
```

```
paramname-0: DeptName
paramtype-0: VARCHAR
paramislist-0: false
paramdesc-0: The department running the expense report
paramname-1: Month
paramtype-1: DATE
paramislist-1: true
paramdesc-1: The month(s) the expense report should cover
```

Creating JavaScript plug-ins

Avaki Studio is the preferred method for creating data services, so it is unlikely that you will have to write a JavaScript plug-in manually. For situations in which you need to perform some kind of operation that isn't supported by Studio and you want a very simple and efficient way to create a data service, a JavaScript plug-in is the preferred choice.

Like Java, JavaScript gives you access to all the Java classes and interfaces available within an Avaki grid server without having to compile or create a JAR file. In fact, the inputs and outputs of your JavaScript plug-in are the same as those for a Java plug-in. For more information about input and output sources, see [“Input and output” on page 177](#).

A JavaScript plug-in consists of a single JavaScript file shared into the Avaki data catalog. The Avaki runtime requires that you define two JavaScript functions: `defineIO()` and `execute()`. The remainder of this section describes how to create a JavaScript plug-in.

Access to Java classes and interfaces

Within your JavaScript plug-in, you have access to any Java class whose package you include via a top-level call to `importPackage()`, as described in [“Import required packages,”](#) below. Refer to [“Creating Java plug-ins with the Plug-in Wizard” on page 183](#) for details on how to use Avaki rowset implementations such as `MemoryRowSet` and `StreamingRowSet`.

Import required packages

You will need to import the package `com.avaki.core.service.dataservice.api`, as well as any Java packages from which you intend to use classes or interfaces (such as `java.sql`, `javax.sql`, and `java.io`). In JavaScript, you do this as follows:

```
importPackage (Package.com.avaki.core.services.dataservice.
    api);
importPackage (Package.java.sql);
importPackage (Package.javax.sql);
importPackage (Package.java.io);

defineIO Function
```

The `defineIO` function consists of a sequence of calls that let the Avaki runtime know about the input sources, parameters, and output results of your plug-in. The Avaki runtime invokes `defineIO()` with an object of type `com.avaki.core.services.dataservice.api.PluginSpec` bound to the variable “`plugin`.” Here is an example of a `defineIO` function:

```
function defineIO()
{
    plugin.addInputStream("InputSource_01", "ResultSet", false,
        "Input stream for: InputSource_01");
    plugin.setOutputStream("OutputStream", "ResultSet",
        "Model output stream");
    plugin.addInputParameter("AccountNumber", Types.VARCHAR,
        false,
        "The account number");
}
```

Methods available on the plug-in object

These methods are available on the plug-in object:

addInputStream

Include one call for each input source that the plug-in expects.

Parameters	Description
InputStream-Name	This is the name to which the input source will be bound to in the execute() method. For example, given the defineIO() function above, the execute() function will have access to an instance of com.avaki.core.services.dataservice.api.InputSource bound to the variable InputSource_01.
Type	The allowed types are ResultSet, XML, and ByteStream.
isList	True or false, depending on whether the parameter is a list of input streams (typically, the value is false for JavaScript plug-ins).
Description	A string description of the input source.

setOutputStream

Include one call to describe the output that the plug-in will produce.

Parameters	Description
OutputStream-Name	This is the name to which the output stream will be bound in the execute() method.
Type	The allowed types are ResultSet, XML, ByteStream.
Description	A string description of the output stream.

addInputParameter

Include once call for each input parameter that your plug-in expects.

Parameters	Description
Name	This is the name to which the input parameter will be bound in the execute method.
Type	The type of the input parameter. This must be one of the types from <code>java.sql.Type</code> .
Description	A string description of the input parameter.

Execute function

The execute function takes no parameters, but all the input sources, the output stream, and the input parameters are bound as global variables when the Avaki runtime invokes it.

```
Function execute()  
{  
  // Your plug-in logic goes here.  
}
```


Provisioning web services

This chapter covers the following topics:

- [“Provisioning web services overview,”](#) below
- [“Setting up the data service”](#) on page 206
- [“Testing data services”](#) on page 214

Provisioning web services overview

Web services are a source of data that can be provisioned into a grid. You can provision data from many kind of web services, such as real-time stock quotes.

Web services are represented by web services description language (WSDL) documents. A WSDL document is a formal statement of the contract between a SOAP client and a SOAP server. It is a platform- and language-independent XML document that describes the format of the SOAP requests expected by the server and the SOAP responses it generates. In addition, the WSDL specifies the name of the web service and the grid server and port where the service can be reached. You can provision third-party web services into an Avaki grid so that the results of the web service operation can be used and transformed by data services.

The sections that follow describe how to use the web UI to provision data from a web service into Avaki. (This can also be done with Avaki Studio; for details, see *Data*

Integration with Sybase Avaki Studio.) As an example, we will provision data from a web service from Google, a third-party web service provider. The web service can be used in a search application. This chapter assumes that you are familiar with the procedure for creating a data service, as described in [Chapter 2, “Basic data integration”](#).

Google’s Web API service is available at the following location:

<http://www.google.com/apis>

To start using the Google web service, you must download the Google Web API Developer’s Kit from this location. The developer’s kit includes a WSDL file that describes the Web service, and examples of accessing the Google Web Service in both Java and VB.NET/C#.

After you download the Google Web API Developer’s Kit, you must create an account with Google. To create an account, go to:

<https://www.google.com/accounts/NewAccount?continue=http://api.google.com/createkey>

Once you create one of these free accounts, you will be assigned a unique license number. This license number must be used whenever a Google Web service method is called. The purpose of this license is to limit the number of calls to the Google Web service to 1,000 invocations per license key per day.

Setting up the data service

To provision a web service and use it in a data service, you must complete the following tasks:

1. Give the data service a name and, optionally, a description.
2. Configure a plug-in for the data service.
3. Configure the data service’s output stream.
4. Configure the data service’s input streams.
5. Choose a grid server for the data service.

The sections that follow provide instructions for these tasks.

Getting started

To begin creating the data service in which the web service will be used, do the following:

- Step 1** Log in as a member of the DataProviders group.
- Step 2** Navigate to the Create Data Service screen:

Home > Data integration > Create data service

The Google search service does not require transformation of data, so choose the data service plug-in option labeled “Use a no-operation plugin that does not transform data.”

Specifying parameters

After you configure a data service plug-in, the screen that appears shows the plug-in that you chose and provides a box where you can specify input streams for the data service. The no-operation plug-in does not require any input parameters, so skip to “Specifying the output stream,” below.

Specifying the output stream

In the Output Stream section, the Name field is already set to the only output stream that is valid for a no-operation plug-in: Result: ByteStream. Skip to “Specifying the input streams,” below.

Specifying the input streams

To specify the input stream, do the following:

- Step 1** Click the Edit Stream link in the box labeled “Primary Input.”

- Step 2** For our Google example, the data for the Primary Input stream will be provided by external web service data. Select “External web service data.”

- Step 3** Click **Next**.

- Step 4** In the URI text box, specify the Uniform Resource Indicator (URI) of the WSDL file that defines the web service that provides data for the stream named Primary Input, or click the **Browse...** button and navigate to a shared WSDL file. Some examples of URIs are HTTP URLs, URLs pointing into your local file system, and URLs of files previously provisioned into Avaki. For our Google example, specify the following URI:

<http://api.google.com/GoogleSearch.wsdl>

Create Data Service

Specify the URI of the WSDL file that defines the web service you'll use to provide data for the stream named Primary Input, then click **Next**. Examples of possible such URIs include HTTP URLs, URLs pointing into your local filesystem, URLs of files previously provisioned into Avaki, and so on. The **Browse...** button below will let you browse Avaki for a WSDL file.

URI:

Use "wrapped" mode for processing document/literal operations

Step 5 (Optional) Place a check mark next to the option “Use wrapped mode for processing document/literal operations” if you want the SOAP binding style to be wrapped, a document literal variation that wraps parameters as children of the root element.

Step 6 Click **Next**.

Step 7 On the Create Data Service screen, select a web service operation to provide the data for the stream’s primary input. The Google Web APIs service package provides the following operations for the Google database:

- doGoogleSearch: perform a Google search and return the results programmatically;
- doGetCachedPage: get access to the cached version of a page from the last time Google’s crawlers last visited it; and
- doSpellingSuggestion: submit a query to the Google Web APIs service and receive in return a suggested spelling correction for the query (if a correction is available).

For our Google example, select the operation doSpellingSuggestion.

Create Data Service

Choose a web service operation to provide the data for the stream Primary Input:

	Service	Port	Binding	Operation
<input type="radio"/>	GoogleSearchService	GoogleSearchPort	GoogleSearchBinding	doGetCachedPage
<input type="radio"/>	GoogleSearchService	GoogleSearchPort	GoogleSearchBinding	doGoogleSearch
<input checked="" type="radio"/>	GoogleSearchService	GoogleSearchPort	GoogleSearchBinding	doSpellingSuggestion

Return the entire SOAP response

Return only the SOAP part

Return only attachment indexed (zero-based):

- Step 8** The following options specify how much of the SOAP response the web service will return:
- **Return the entire SOAP response:** The web service returns the SOAP message and any attachments.
 - **Return only the SOAP part:** The web service returns the SOAP message but not any attachments.
 - **Return only attachment indexed (zero-based):** Return a specified attachment but not the SOAP message. If the web service returns multiple attachments, you can enter a number in the text box to specify the zero-based index number of the attachment to return. For example, if the web service returns four attachments but you only want the fourth, enter 3 in the text box.

For our Google example, select “Return the entire SOAP response.”

Step 9 Click **Next**.

Step 10 The screen that appears shows the parameters that the input stream requires. In our Google example, the web service operation doSpellingSuggestion requires two AROMValue input stream parameters of type string. AROMValue is a proprietary Avaki data structure.

Create Data Service

No Parameters	<div style="border: 1px solid gray; padding: 5px; margin-bottom: 5px;"> <p style="text-align: center;">Input Stream</p> <p>Name: Primary Input Stream: ByteStream Type: From: Web Service Operation Target: Service: GoogleSearchService, Port: GoogleSearchPort, Binding: GoogleSearchBinding, Operation: doSpellingSuggestion, Endpoint: http://api.google.com/search/beta2 Edit Stream</p> </div> <div style="border: 1px solid gray; padding: 5px; margin-bottom: 5px;"> <p style="text-align: center;">Input Stream Parameter</p> <p>Name: (0) key Type: AROMValue Description: Parameter 0: string Value: <key xsi:type="xsd:string" xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"/> Edit Parameter</p> </div> <div style="border: 1px solid gray; padding: 5px;"> <p style="text-align: center;">Input Stream Parameter</p> <p>Name: (1) phrase Type: AROMValue Description: Parameter 1: string Value: <phrase xsi:type="xsd:string" xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"/> Edit Parameter</p> </div>	<div style="border: 1px solid gray; padding: 5px; margin-bottom: 5px;"> <p style="text-align: center;">Plug-in</p> <p>Name: No Operation (NoOp) Description: Passes the primary input to the output without modification Coherence: 5 minutes Window: Change Plug-in No Input Parameters</p> </div> <div style="border: 1px solid gray; padding: 5px;"> <p style="text-align: center;">Output Stream</p> <p>Name: Result: ByteStream <input type="text"/></p> </div>
---------------	---	--

For each required parameter, you must specify how the value for the stream parameter will be obtained. For our Google example, do the following:

- Click on the Edit Parameter link for the input parameter named “key.”

Create Data Service

Specify how the value for the stream parameter should be obtained.

Parameter Name: (0) key
 Type: AROMValue
 Value: Use this value:

Field Name	Null	Value
• key [string]:		<input style="width: 90%;" type="text"/>

Use an existing parameter to the Data Service.
(There are no data service parameters of type AROMValue.)

Create a new AROMValue parameter to the data service.
 Name:
 Description:

- For each parameter, you can select one of the following to specify how the value for the stream parameter should be obtained:
 - **Use this value:** Specify a static value to use whenever the data service is executed.
 - **Use an existing parameter to the data service:** This option is available only if there is already an existing dynamic parameter in the data service. Any existing parameters appear on the top left of the Create Data Service screen.
 - **Create a new AROMValue parameter to the data service:** To create a new dynamic AROMValue parameter, specify a name and description for a value that the user will specify when the data service is executed.

For our Google example, select “Use this value.” In the Value text box, enter the license key that you received from Google. For more about the other options, see [“On AROMValues as runtime parameters”](#) below.

- Click **Submit**.

- Click on the Edit Parameter link for the input parameter named “phrase.”

Update Data Service

Specify how the value for the stream parameter should be obtained.

Parameter Name: (1) phrase
 Type: AROMValue
 Value: Use this value:

Field Name	Null	Value
• phrase [string]:		<input style="width: 100%;" type="text"/>

Use an existing parameter to the Data Service.
(There are no data service parameters of type AROMValue.)

Create a new AROMValue parameter to the data service.
 Name:
 Description:

- For our Google example, select **Use this value** to specify how the value for the stream parameter should be obtained. In the Value text box, enter the word or phrase to for which you want Google to return a suggested spelling correction, such as “spinaker.”
- Click **Submit**.
- On the screen that appears, click **Next**.

On AROMValues as runtime parameters

For most data service parameters, the value you’d supply under “Use this value” is the same as that you’d be prompted for at runtime if you associated the stream or plug-in parameter with a data service parameter. For AROMValues, however, this is not the case. Even a simple AROMValue (like “phrase”, above) is transformed by Avaki from the simple value you’d enter in the text box to a lengthy fragment of XML; but that only happens when you enter a fixed “Use this value” value. If you make the parameter dynamic by associating it with a data service parameter, it’s that lengthy fragment of XML that must be entered manually at runtime.

This limitation does not apply to web service data services defined in Avaki Studio. Therefore, if you want your web service data service to have dynamic parameters, you’re usually best off defining that data service in Studio. For more information on Studio and its use of web service data, see *Data Integration with Sybase Avaki Studio*.

Specifying a grid server

On the screen that appears, enter a name for the data service and choose the grid server where you would like to create it. Specify a value for the expiration of the data service's results and the coherence window for the data service's plug-in.

Create Data Service

Choose a name for the data service and the server where you would like to create it. You may also specify a value for the expiration of the data service's data and for the coherence window for the data service's plug-in. When you are done, click **Next** to create the data service. You may also click **Export Descriptor** to save the current state of this data service as a data service descriptor file and continue creating it at a later time.

Or click **Return to Overview** to return to editing the name, description, parameters, input stream(s), plug-in, or output type of this data service.

Data service name:

Description:

Grid server: ▼

Cached data expiration: No caching
 Never expires
 Expires after ▼

Plug-in coherence window: ▼

Execute data service as: The user calling the data service
 A specific user:
(e.g. `userName@authService.authServiceType.domain`)

- Step 1** Enter a name for this data service.
- Step 2** (Optional) Enter a description of this data service.
- Step 3** From the grid server pull-down menu, select a grid server for this data service.
- Step 4** Cached data expiration: Select one of the following to indicate whether the data is cached and, if so, when the data expires from the cache:
- No caching: The data is not cached.
 - Never expires: The data never expires from the cache.
 - Expires after n seconds: Specify the interval (in seconds, minutes, hours, or days) before the data expires from the cache.

Step 5 In the plug-in coherence window field, specify the duration (in seconds, minutes, hours, or days) during which the contents of the data service are assumed to be fresh after the cache service has last inspected the back-end source object for updates. The default is 5 minutes.

Step 6 Select the user to execute the data service as: the user calling the service or a specific user. If you choose to execute the data service as a specific user, specify the qualified user name of a grid user who has permission to execute this database service. Use the following format:

```
<user-name>@<authservice>.<authservicetype>.<domain>
```

For example:

```
wilma@DefaultAuthService.Grid.Bedrock
```

Step 7 Click **Next** or **Export Descriptor** to finish creating the data service. The View Data Services screen appears, listing the new data service.

Testing data services

Follow these steps to test whether the data service is operational:

Step 1 Navigate to the View Data Services screen:

Home > Data integration > Manage data services

View Data Services

Grid domain: ▼

You are viewing the data services in the current domain.

Name	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata
MyDataService	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata
MyWebDS	View/Edit	Test	Schedule	Remove	Attributes	Security	Metadata

- Step 2** Click the **Test** link beside the data service you want to test. The Test Data Service screen appears.

Test Data Service

Specify the values of the input parameters for this data service and then click **Submit** to execute the service.

Name	Type	Description	Value
GridFileTest	GridFile	DSTest	<input style="width: 100%;" type="text"/>

- Step 3** If the data service required parameters, you would specify the values in the Value box. Our Google example does not require input parameters, so click **Submit**. The system displays a message indicating whether the data service is operational. For our Google example, the system returns the following output, which includes a suggestion for the correct spelling of “spinnaker.”

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
- <SOAP-ENV:Body>
  - <ns1:doSpellingSuggestionResponse xmlns:ns1="urn:GoogleSearch" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <return xsi:type="xsd:string">spinnaker</return>
  </ns1:doSpellingSuggestionResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


Managing views

This chapter explains how to set up, modify, and run Avaki views. A view consists of a *view generator* and a *generated view* file. The view generator gets input from a file, a database operation or a data service, transforms the data, then saves the results as a generated view file.

View generators create files that typically represent the output of a data service or a database operation. View generators come in three types: those that take input data from data services, those that take input data from database operations, and those that take input data from files. You can configure a view generator simply to pass its input untransformed into the output file; to transform rowset input to CSV or HTML format; or to apply an XSLT style sheet to the input to produce XML output. (If you wish to transform a large data set into an output format that isn't supported by Avaki, XSLT might not be the right option because the result set might not fit in memory. In those cases, you can implement the transformation logic you need in a Java class that implements the TrAX interface; see [“Non-XSLT-based view generators” on page 242.](#))

A generated view is a transactionally consistent snapshot of the results of its input. That is, if a view generator is running while a user or program is reading its generated view, that user or program continues to read the version that they were reading. At the same time, however, a new snapshot appears in the data catalog in place of the old one. Anyone who attempts to read the view after the run has completed sees the new snapshot. When generated views are unused for a certain period, they are discarded.

In this chapter:

- [“Managing view generators” on page 218](#)

- [“Generating views” on page 240](#)
- [“Non-XSLT-based view generators” on page 242](#)

Managing view generators

This section covers the following topics:

- [“Setting up file view generators” on page 218](#)
- [“Setting up database operation view generators” on page 221](#)
- [“Setting up data service view generators” on page 225](#)
- [“Modifying view generators” on page 229](#)

Setting up file view generators

A file view generator:

- extracts data from a file;
- transforms the data, if configured to do so; and
- stores the data in a generated view file.

Follow these steps to set up a file view generator:

Step 1 Log in to Avaki as a member of the DataProviders group.

Step 2 Navigate to the Create View Generator screen:

Home > Data integration > Create view generator

Choose View Source Type

Indicate the kind of data source on which you want to base your view, then click **Continue** to choose a source.

File
 Database operation
 Data service









Step 3 Select **File**.

Step 4 Click **Continue**. The Browse Directories screen appears.

Browse Directories

The top-level directory in a grid domain is /. Underneath is the /System subdirectory, which contains directories and files related to a particular grid domain, and the /Interconnects link to the Interconnects directory, which contains the root directories of any domains to which a domain is interconnected. A grid domain's administrator can view and modify the settings for an interconnected domain if the administrator has been granted access rights for modifying the domain.

You can add an Avaki directory at any level if you have write permission for the directory in which you want to create a directory. By default, only the Avaki administrator can create new directories at the top level, /.

<input type="checkbox"/>		/		Attributes	Security
<input type="checkbox"/>		GeneratedViews		Attributes	Security
<input type="checkbox"/>		Interconnects		Attributes	Security
<input type="checkbox"/>		Metadata		Attributes	Security
<input type="checkbox"/>		Shares		Attributes	Security
<input type="checkbox"/>		System		Attributes	Security
<input type="checkbox"/>		WSDLs		Attributes	Security
<input type="checkbox"/>		Categories		Attributes	Security

Step 5 Click on directory names (Shares, for example) to navigate to the file for which you want to create a view generator. When you reach the file, click the box to its left to select it.

- Step 6** Near the bottom of the screen, click **Create View Generator**. The Create View screen appears.

Create View

Please specify the following settings for the view, then click **Submit** to create the view.

Input file path: /Shares/Water Buffalo Lodge/secrethandshake.txt

Output file name:

Output directory: [Browse](#)

Description (optional):

Output format: Unmodified
 Style Sheet

Engine:

Path: [Browse](#)

Parameters: [Set Output Parameters](#)

Grid Server:

- Step 7** In the Output file name field, enter a name for the generated view file that this view generator will produce.
- Step 8** In the Output directory field, leave the default directory in place, or enter or browse for a grid directory in which to place the generated view file.
- Step 9** In the Description field, enter a description of this view generator. (This is recommended.)
- Step 10** In the Output format field, select an output format for the generated view. If you select Unmodified, skip to [Step 12](#).
- Step 11** If you selected the Style Sheet output format, fill in the Engine, Path, and Parameters fields:
- **Engine:** Select a style sheet engine to use for the transform. You can select Saxon or Xalan, the style sheet engines provided by Avaki, or you can provide your own engine. If you are providing your own style sheet engine, set the pull-down to Other and enter your engine's TransformerFactoryImpl class name in the field to the right. You can find the name in your XSLT processor's JAR files. Here are some common ones:
 - Saxon 6.5.2: com.icl.saxon.TransformerFactoryImpl
 - Saxon 7: net.sf.saxon.TransformerFactoryImpl

- Xalan: org.apache.xalan.processor.TransformerFactoryImpl
- jd.xslt: jd.xml.xslt.trax.TransformerFactoryImpl
- Path: Enter or browse for the grid path to the style sheet you want to use.
- Parameters: If your style sheet requires output parameters, click the link [Set Output Parameters](#). The Enter XSLT Parameters screen appears. On it, you can enter name and value pairs of parameters. Click **Submit** or **Done** to return to the Create View screen.

Step 12 From the Grid server pull-down list, select a grid server for this view generator.

Step 13 Click **Submit** to create the view generator. The system displays a confirmation page that shows details pertaining to the new view generator.

If you're ready to run the view generator, click **Test**. The system attempts to run the view generator and displays a screen reporting success or failure. **Note:** Do not attempt to test your view generator yet if you are providing your own style sheet engine ([Step 11](#) above).

Step 14 If you are providing your own style sheet engine, do the following to hot-deploy the JAR file for your XSLT processor:

- Log in to the machine on which the view generator will run.
- Put the JAR file in this directory:

```
<Avaki-install-dir>/jboss/server/grid-server/deploy
```

Setting up database operation view generators

A database operation view generator:

- calls a database operation and receives its results,
- transforms the data, if configured to do so, and
- stores the data in a generated view file.

Note Before you can set up a database operation view generator, you must create a database connector and a database operation as described in [Chapter 1](#), “[Managing information from databases](#)”.

Follow these steps to set up a database operation view generator:

Step 1 Log in to Avaki as a member of the DataProviders group.

Step 2 Navigate to the Create View Generator screen:

Home > Data integration > Create view generator

Choose View Source Type

Indicate the kind of data source on which you want to base your view, then click **Continue** to choose a source.

File
 Database operation
 Data service

Step 3 Select Database operation.

Step 4 Click **Continue**. The Create Database View Generator screen appears.

Create Database View Generator

You have chosen to create a new database view generator. Please select the database operation to use.

Grid domain: ▼

Select	Connector	Name
<input type="radio"/>	MyDBConnector	MyDBOperation

- Step 5** Click in the Select column to choose a database operation for your view generator, then click **Continue**. The Create View screen appears.

Create View

Specify the following settings for the view:

Connection name: MyDBCconnector
 Operation name: MyDBOperation
 SQL parameters: Specify manually
 Obtain from the following XML input file:
 [Browse](#)

Output file name:

Output directory: /GeneratedViews [Browse](#)

Description (optional):

Run view as: The user running the view
 A specific user:
 (e.g. `userName@authService.authServiceType.domain`)

Output format: Unmodified
 CSV
 HTML
 Style Sheet

Engine:

Path: [Browse](#)

Parameters [Set Output Parameters](#)

Grid Server:

- Step 6** If the SQL parameters required by your database operation are in a file, select “Obtain from the following XML input file” in the SQL parameters field. Then click the Browse link, find the file, and select it. (When you finish selecting the file, you’ll be returned to this Create View screen.)
- Step 7** In the Output file name field, enter a name for the generated view file that this view generator will produce.
- Step 8** In the Output directory field, leave the default directory in place, or enter or browse for a grid directory in which to place the generated view file.
- Step 9** In the Description field, enter a description of this view generator. (This is recommended.)

- Step 10** In the Run view as area, specify which user the view will be run as. To run the view as the current user, click **The user running the view**. To run the view as a specific user, click **A specific user** and enter the qualified user name. Use the following format:

```
<user-name>@<authservice>.<authservicetype>.<domain>
```

For example:

```
wilma@DefaultAuthService.Grid.Bedrock
```

Note The Avaki web UI employs a special browsing feature when you're selecting the run-as user for a data service, database operation, or view. To make user browsing work properly, make sure the smooth scrolling option in your web browser is turned off. In Firefox or Internet Explorer, select Tools > (Internet) Options > Advanced and uncheck "Use smooth scrolling."

- Step 11** In the Output format field, select an output format for the generated view. If you do *not* select Style Sheet, skip to [Step 13](#).
- Step 12** If you selected the Style Sheet output format, fill in the Engine, Path, and Parameters fields:
- **Engine:** Select a style sheet engine to use for the transform. You can select Saxon or Xalan, the style sheet engines provided by Avaki, or you can provide your own engine. If you are providing your own style sheet engine, set the pull-down to Other and enter your engine's TransformerFactoryImpl class name in the field to the right. You can find the name in your XSLT processor's JAR files. Here are some common ones:
 - Saxon 6.5.2: com.icl.saxon.TransformerFactoryImpl
 - Saxon 7: net.sf.saxon.TransformerFactoryImpl
 - Xalan: org.apache.xalan.processor.TransformerFactoryImpl
 - jd.xslt: jd.xml.xslt.trax.TransformerFactoryImpl
 - **Path:** Enter or browse for the grid path to the style sheet you want to use.
 - **Parameters:** If your style sheet requires output parameters, click the link Set Output Parameters. The Enter XSLT Parameters screen appears. On it, you can enter name and value pairs of parameters. Click **Submit** or **Done** to return to the Create View screen.
- Step 13** In the Grid server pull-down, select a grid server for this view generator.

Step 14 Click **Continue**. The Enter View Parameters screen appears. If your database operation requires input parameters, enter them here.

Step 15 Click **Submit** to create the view generator. The system displays a confirmation page that shows details pertaining to the new view generator.

If you're ready to run the view generator, click **Test**. The system attempts to run the view generator and displays a screen reporting success or failure. **Note:** Do not attempt to test your view generator yet if you are providing your own style sheet engine ([Step 12](#) above).

Step 16 If you are providing your own style sheet engine, do the following to hot-deploy the JAR file for your XSLT processor:

- Log in to the machine on which the view generator will run.
- Put the JAR file in this directory:

```
<Avaki-install-dir>/jboss/server/grid-server/deploy
```

Setting up data service view generators

A data service view generator:

- calls an Avaki data service and receives its results,
- transforms the data, if configured to do so, and
- stores the data in a generated view file.

Note Before you can set up a data service view generator, you must create a data service as described in [“Basic data integration”](#) on page 49.

Follow these steps to set up a data service view generator:

Step 1 Log in to Avaki as a member of the DataProviders group.

Step 2 Navigate to the Create View Generator screen:

Home > Data integration > Manage view generators

Choose View Source Type

Indicate the kind of data source on which you want to base your view, then click **Continue** to choose a source.

File
 Database operation
 Data service

Step 3 Select Data service.

Step 4 Click **Continue**.

Step 5 Select the data service to use.

Step 6 Click **Continue**. The Create View screen appears.

Create View

Specify the following settings for the view. Then click **Continue** to enter the data service parameters for the view.

Data service: MyDataService

Output file name:

Output directory:

Description (optional):

Run view as: The user running the view
 A specific user:
 (e.g. *userName@authService.authServiceType.domain*)

Output format: Unmodified
 CSV
 HTML
 Style Sheet

Engine:

Path:

Parameters [Set Output Parameters](#)

Grid Server:

Step 7 In the Output file name field, enter a name for the generated view file that this view generator will produce.

- Step 8** In the Output directory field, leave the default directory in place, or enter or browse for a grid directory in which to place the generated view file.
- Step 9** In the Description field, enter a description of this view generator. (This is recommended.)
- Step 10** In the Run view as area, specify which user the view will be run as. To run the view as the current user, click **The user running the view**. To run the view as a specific user, click **A specific user** and enter the qualified user name. Use the following format:

```
<user-name>@<authservice>.<authservicetype>.<domain>
```

For example:

```
wilma@DefaultAuthService.Grid.Bedrock
```

Note The Avaki web UI employs a special browsing feature when you're selecting the run-as user for a data service, database operation, or view. To make user browsing work properly, make sure the smooth scrolling option in your web browser is turned off. In Firefox or Internet Explorer, select Tools > (Internet) Options > Advanced and uncheck "Use smooth scrolling."

- Step 11** In the Output format field, select an output format for the generated view. If you do *not* select Style Sheet, skip to [Step 13](#).
- Step 12** If you selected the Style Sheet output format, fill in the Engine, Path, and Parameters fields:
- Engine: Select a style sheet engine to use for the transform. You can select Saxon or Xalan, the style sheet engines provided by Avaki, or you can provide your own engine. If you are providing your own style sheet engine, set the pull-down to Other and enter your engine's TransformerFactoryImpl class name in the field to the right. You can find the name in your XSLT processor's JAR files. Here are some common ones:
 - Saxon 6.5.2: com.icl.saxon.TransformerFactoryImpl
 - Saxon 7: net.sf.saxon.TransformerFactoryImpl
 - Xalan: org.apache.xalan.processor.TransformerFactoryImpl
 - jd.xml: jd.xml.xslt.trax.TransformerFactoryImpl
 - Path: Enter or browse for the grid path to the style sheet you want to use.
 - Parameters: If your style sheet requires output parameters, click the link Set Output Parameters. The Enter XSLT Parameters screen appears. On it, you can enter name

and value pairs of parameters. Click **Submit** or **Done** to return to the Create View screen.

- Step 13** In the Grid server pull-down, select a grid server for this view generator.
- Step 14** Click **Continue**. The Enter Data Service View screen appears. If your data service requires input parameters, enter them here.
- Step 15** Click **Submit** to create the view generator. The system displays a confirmation page that shows details pertaining to the new view generator.

If you're ready to run the view generator, click **Test**. The system attempts to run the view generator and displays a screen reporting success or failure. **Note:** Do not attempt to test your view generator yet if you are providing your own style sheet engine (Step 12 above).

- Step 16** If you are providing your own style sheet engine, do the following to hot-deploy the JAR file for your XSLT processor:
- Log in to the machine on which the view generator will run.
 - Put the JAR file in this directory:

```
<Avaki-install-dir>/jboss/server/grid-server/deploy
```

Displaying view dependencies

To display a list of any database operations that a view depends on or data services that depend on a view, do the following:

- Step 1** Navigate to the Display View Generators screen:

Home > Data integration > Manage view generators

Display View Generators										
Grid domain: <input type="text" value="CheryDomain"/>										
You are viewing the view generators in the current domain.										
Name	Creation Date	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
fedsform.txt	Wed Oct 06 05:44:01 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
gazoo-vu	Wed Oct 06 05:44:23 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
newhandshake.txt	Wed Oct 06 05:44:54 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
<input type="button" value="Done"/>										

The system displays a list of the views in the current grid domain.

Step 2 Click the **Dependencies** link beside the name of the view.

Dependencies for /System/Domains/CherylDomain/Views/gazoo-vu--2004_10_06__05_44_23_0424_EDT		
Operation	Dependency	Dependency Type
Data Service	CherylDomain.MyDataService	Input to /System/Domains/CherylDomain/Views/gazoo-vu--2004_10_06__05_44_23_0424_EDT

The system displays a list of database operations that the view depends on or data services that depend on the view.

Modifying view generators

The procedures in this section explain how to modify view generators of all types. You can perform any of the following tasks:

- [“Editing a view generator,”](#) below
- [“Scheduling updates for views”](#) on page 232
- [“Configuring update notifications for views”](#) on page 238

Editing a view generator

Follow these steps to change the description or output format for a view generator.

Step 1 Log in to Avaki as a member of the DataProviders group.

Step 2 Navigate to the Display View Generators screen:

Home > Data integration > Manage view generators

Display View Generators

Grid domain: CheryDomain

You are viewing the view generators in the current domain.

Name	Creation Date	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
fredsform.txt	Wed Oct 06 05:44:01 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
gazoo-vu	Wed Oct 06 05:44:23 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
newhandshake.txt	Wed Oct 06 05:44:54 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies

- Step 3** Click the Edit link for the view generator you want to change. The View/Edit View screen appears.

View/Edit View

Specify the following settings for the view:

Data service: MyDataService
 Output file name: gazoo-vu
 Output directory: /GeneratedViews

Description:

Run view as: The user running the view
 A specific user:
(e.g. userName@authService.authServiceType.domain)

Output format: Unmodified
 CSV
 HTML
 Style Sheet

Engine

Path [Browse](#)

Parameters [Set Output Parameters](#)

- Step 4** You can edit the text in the Description field, change the user that the view is run as, and change the output format. If the Style Sheet output format is selected, you can also change the stylesheet engine, add or change the path, and change the output parameters (by clicking on the link).
- Step 5** Click **Continue** when you're done modifying the view generator. The Update View Parameters screen appears.
- Step 6** If your view generator includes parameters, you can modify them.
- Step 7** Click **Submit** when you're done. The system saves the changes to your view generator and displays a confirmation page.

Scheduling updates for views

When you schedule a view, you specify the interval at which the view generator will run and the time at which the schedule will start. For example, you might schedule your view to run every 90 minutes starting now, or every seven days starting next Monday. You can configure multiple schedules for each view generator.

Follow these steps to schedule a view generator.

Step 1 Log in to Avaki as a member of the DataProviders group.

Step 2 Navigate to the Display View Generators screen:

Home > Data integration > Manage view generators

Display View Generators

Grid domain: CheryDomain

You are viewing the view generators in the current domain.

Name	Creation Date	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
fredform.txt	Wed Oct 06 05:44:01 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
gazoo-vu	Wed Oct 06 05:44:23 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
newhandshake.txt	Wed Oct 06 05:44:54 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies

Step 3 Click the Schedule link for the view generator you want to change. The Show View Schedules screen appears.

Show View Schedules

Your view has no schedules at present.

Step 4 Click the **Add Schedule** button. The Add New Schedule screen appears.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

One time
 Periodic
 Calendared
 Advanced

Starting Now
 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur every: minutes ▼

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Schedule exclusions:

No schedule exclusions are currently defined in this domain.

Step 5 Click a tab to choose the type of schedule: One time, Periodic, Calendared, or Advanced. The Advanced and Calendared options are similar. The Advanced option lets you use a cron expression—powerful but cryptic—to schedule the recurrence interval. The Calendared option offers a friendlier interface to a subset of the functionality enabled by cron expressions.

Step 6 Go to the appropriate procedure to complete your schedule entry:

- [“Configuring one-time view generation schedules,”](#) below
- [“Configuring calendared view generation schedules”](#) on page 235
- [“Configuring periodic or advanced view generation schedules”](#) on page 237

Configuring one-time view generation schedules. Follow the steps in [“Scheduling updates for views” on page 232](#) before starting this subprocedure.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

One time
 Periodic
 Calendared
 Advanced

Do once: Now

12 AM :00 Jan 1 2005

Follow these steps:

- Step 1** Choose a time zone for this schedule. The schedule can be specified according to the grid server’s time zone or relative to Greenwich Mean Time (GMT).
- Step 2** In the Do once field:
- If you want the one-time view generation to occur immediately, click the **Now** button.
 - If you want the view generation to occur later, click the lower button. Then use the pull-down menus to select the time, month, day and year.
- Step 3** Click **Submit** to save your schedule entry. The system displays the new entry on the Show Pin Schedules screen.

Note For instructions on setting up schedule exclusions—specific times when the view is not generated according to the schedule—see [“Configuring schedule exclusions” on page 166](#).

Configuring calendared view generation schedules. Follow the steps in “[Scheduling updates for views](#)” on page 232 before starting this subprocedure.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

One time
 Periodic
 Calendared
 Advanced

Now
After 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur at: 12 AM ▼ :00 ▼ on:

Days	Months	Years
<input checked="" type="radio"/> all <input type="radio"/> of week <input type="radio"/> of month <input type="radio"/> of week in month	Jan ▲ Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec all ▼	2004 ▲ 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 all ▼

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Follow these steps:

- Step 1** Choose a time zone for this schedule. The schedule can be specified according to the grid server’s time zone or relative to Greenwich Mean Time (GMT).
- Step 2** In the After field, specify when this schedule entry takes effect:
- If you want the one-time view generation to occur immediately, click the **Now** button.
 - If you want the view generation to occur later, click the lower button. Then use the pull-down menus to select the time, month, day and year.

- Step 3** In the **Recur at** field, use the pull-down menus to specify the time of day at which you want the view generation to take place. (If you want the view generation to occur more than once a day, you can use a periodic or advanced schedule, or you can create separate schedule entries for the other view generations.)
- Step 4** In the **Days** column, choose how you want to specify days in this schedule entry:
- **All:** every day.
 - **Of week:** Sunday through Saturday—click one or more days.
 - **Of month:** 1, 2, 3...—click one or more days.
 - **Of week in month:** use the pull-down menus to choose the first, second, third, fourth, fifth, or last occurrence of any day of the week (the first Monday, for example).
- Step 5** In the **Months** column, select one or more months during which this schedule entry will be in effect, or select **all** for all months. Use Shift-click or Control-click to select multiple months.
- Step 6** In the **Years** column, select one or more years during which this schedule entry will be in effect, or select **all** for all years. Use Shift-click or Control-click to select multiple years.
- Step 7** In the **Continue recurring** field, specify how long you want this schedule entry to remain in effect: forever, for a specified number of refreshes, or until a specified date and time.
- Step 8** Click **Submit** to save your schedule entry. The system displays a summary of the new entry, including the time of next execution, on the **Show Pin Schedules** screen.

Note For instructions on setting up schedule exclusions—specific times when the view is not generated according to the schedule—see [“Configuring schedule exclusions” on page 166](#).

Configuring periodic or advanced view generation schedules. Follow the steps in [“Scheduling updates for views” on page 232](#) before starting this subprocedure.

Add New Schedule

Select a time zone for this schedule: (server's local time zone) ▼

Type of schedule:

Now
 After 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur according to this cron expression:

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Starting Now
 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Recur every: minutes ▼

Continue recurring

forever
 at most times
 until 12 AM ▼ :00 ▼ Jan ▼ 1 ▼ 2005 ▼

Follow these steps:

- Step 1** Choose a time zone for this schedule. The schedule can be specified according to the grid server's time zone or relative to Greenwich Mean Time (GMT).
- Step 2** In the After field, specify when this schedule entry takes effect:
- If you want the one-time view generation to occur immediately, click the **Now** button.

- If you want the view generation to occur later, click the lower button. Then use the pull-down menus to select the time, month, day and year.

Step 3 Use the Recur... field to specify the interval at which this schedule is executed:

- If you're creating a periodic schedule entry, enter an integer and select from the pull-down to specify an interval—for example, every 40 minutes, every 5 days, or every 2 months.
- If you're creating an advanced schedule entry, you must enter a cron expression of this form in the Recur... field:

```
<seconds> <minutes> <hours> <days-of-month> <months>
<days-of-week> [<years>]
```

See the *Sybase Avaki EII Command Reference* for details of the cron syntax.

Step 4 In the Continue recurring field, specify how long you want this schedule entry to remain in effect: forever, for a specified number of refreshes, or until a specified date and time.

Step 5 Click **Submit** to save your schedule entry. The system displays a summary of the new entry, including the time of next execution, on the Show View Schedules screen.

Note For instructions on setting up schedule exclusions—specific times when the view is not generated according to the schedule—see [“Configuring schedule exclusions” on page 166](#).

Configuring update notifications for views

You can configure a view generator (call it fredform) to issue a notification every time it updates its generated view. The notification enables other views that rely on fredform to regenerate their generated views, keeping their data current. This feature provides notification on change only.

Note If you want to send update notifications to another grid domain or receive update notifications from another grid domain, the two grid domains must be interconnected and cross-domain messaging must be configured for both domains. See the *Sybase Avaki EII Administration Guide* for instructions on setting up cross-domain messaging.

Follow these steps to configure update notifications from one view generator to another.

Step 1 Log in to Avaki as a member of the DataProviders group.

Step 2 Navigate to the Display View Generators screen:

Home > Data integration > Manage view generators

Display View Generators										
Grid domain: <input type="text" value="CheryDomain"/>										
You are viewing the view generators in the current domain.										
Name	Creation Date	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
fiefsform.txt	Wed Oct 06 05:44:01 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
gazoo-vu	Wed Oct 06 05:44:23 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
newhandshake.txt	Wed Oct 06 05:44:54 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
<input type="button" value="Done"/>										

Step 3 Click the Notifications link for the view generator that will receive the update notifications. The Update View Notification List screen appears.

Update View Notification List		
	Notification Source	Delete
New:	<input type="text"/>	Browse
<input type="button" value="Submit"/> <input type="button" value="Cancel"/>		

Step 4 Click the Browse link. On the Browse Directories screen, navigate to the view file whose changes will kick off an update notification. (You can select either a generated view or a view generator.)

Step 5 When you find the file, click the button to the left of the file name to select it, then click **Continue**. The system redisplay the Update View Notification List showing the file you selected as a notification source.

Step 6 Click **Submit** to save the notification list.

Deleting views

Follow these steps to delete a view generator.

Step 1 Log in to Avaki as a member of the DataProviders group.

Step 2 Navigate to the Display View Generators screen:

Home > Data integration > Manage view generators

Display View Generators										
Grid domain: CheryDomain										
You are viewing the view generators in the current domain.										
Name	Creation Date	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
fredsform.txt	Wed Oct 06 05:44:01 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
gazoo-vu	Wed Oct 06 05:44:23 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
newhandshake.txt	Wed Oct 06 05:44:54 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
<input type="button" value="Done"/>										

Step 3 Click the Remove link for the view generator you want to delete.

Step 4 Click **OK** in the confirmation window. The system removes the view generator and redisplay the Display View Generators screen without it.

Generating views

You can run a view generator manually, as described in this section, or you can schedule the view to run automatically, as described in [“Scheduling updates for views” on page 232](#).

About generated view files

Each time a view generator runs, it replaces the existing generated view with a new file. Do not rename or delete a generated view. If you want to change the location where a view will be generated, you must reconfigure the view generator.

Similarly, do not create hard links to generated views; these hard links will no longer function when a view is updated and the generated view is replaced. If you need to create a link to a generated view, use the **avaki ln -s** command in the CLI to create a soft link.

Cache interactions

Like other files in the data catalog, the files used by file-based view generators (input files and style sheets) may be cached. If you modify an input file or style sheet but see no corresponding change in the generated view when you run your view generator, it's likely that the view generator is using cached copies of one or both files. To correct the problem, invalidate the cached copies of the input file and style sheet (see [“Invalidating cached items” on page 136](#)) and run the view generator again.

View generation procedure

Follow these steps to run a view generator and update its generated view.

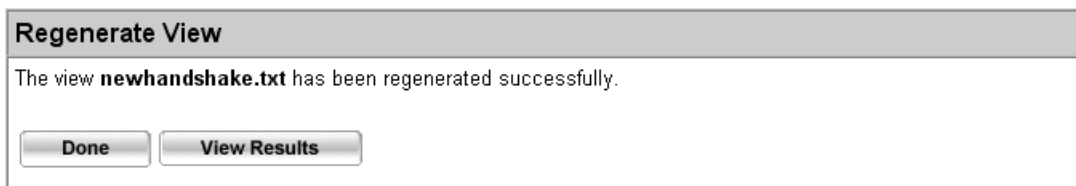
Step 1 Log in to Avaki as a member of the DataProviders group.

Step 2 Navigate to the Display View Generators screen:

Home > Data integration > Manage view generators

Display View Generators										
Grid domain: <input type="text" value="CheryDomain"/>										
You are viewing the view generators in the current domain.										
Name	Creation Date	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
fiefsform.txt	Wed Oct 06 05:44:01 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
gazoo-vu	Wed Oct 06 05:44:23 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
newhandshake.txt	Wed Oct 06 05:44:54 EDT 2004	Edit	Schedule	Notifications	Run	Output	Remove	Attributes	Security	Dependencies
<input type="button" value="Done"/>										

- Step 3** Click the **Run** link for the view generator you want to run. The Regenerate View screen appears.



- Step 4** (Optional.) Click the **View Results** button to display the results (the contents of the generated view file).

Non-XSLT-based view generators

Data integration and complex transformation logic should be deployed in data services. You can use Avaki Studio to create and deploy data services (see *Data Integration with Sybase Avaki Studio*), or you can write your own data services, as described in [Chapter 4, “Setting up data service plug-ins”](#). Use the method described in this section only if you can’t accomplish what you need using data services.

In some situations, the transformation options offered by standard Avaki view generators may not be sufficient. In particular, if you wish to transform a large data set to an output format that isn’t supported by Avaki, XSLT might not be the right option because the result set might not fit in memory. In those situations, you can implement the transformation logic that you need in a Java class that implements the TrAX interface, using a SAX filter to perform the transformation function in a scalable manner. TrAX transformers accept XML input from a single source (no integration) and allow integration with third-party applications.

This section describes how to implement and deploy TrAX transformers for view generators in Avaki. These topics are covered:

- [“The TrAX standard,”](#) below
- [“Implementing a Java transformer”](#) on page 243
- [“Installing your Java transformer”](#) on page 245
- [“Referring to other documents in your transformer”](#) on page 245
- [“Logging errors”](#) on page 246

The TrAX standard

TrAX (Transformation API for XML) is a collection of APIs for transforming XML files. TrAX is a subset of the JAXP (Java API for XML Processing) standard.

Use the TrAX API classes defined in JRE 1.4.1. For more information on the TrAX API, see:

<http://java.sun.com/j2se/1.4.1/docs/api/javax/xml/transform/package-summary.html>

Implementing a Java transformer

To create a Java formatter you must implement the following TrAX interfaces:

```
javax.xml.transform.TransformerFactory
javax.xml.transform.Templates
javax.xml.transform.Transformer
```

You can write Java classes to implement these interfaces from scratch. To make it easier for you to implement the interfaces, Avaki provides some base classes that you can extend to make the required TrAX classes. These are:

```
com.avaki.trax.AbstractTransformer
com.avaki.trax.AbstractTransformerFactory
```

The `Templates` interface is relatively simple, so Avaki does not provide an extensible base class. You can find these classes in `<Avaki-install-dir>/examples/trax/avaki_trax.jar`. Put this JAR in your classpath when you compile.

There is a simple example of a custom Java transformer in `<Avaki-install-dir>/examples/trax`.

The Transformer class

When you extend `AbstractTransformer`, your subclass must include a method like this:

```
public void transform(Source xmlSource, Result outputTarget)
    throws TransformerException
{
    // TO DO:
    // Add your Java code here to transform xmlSource
    // as you like.
}
```

Note The `AbstractTransformer` class includes two methods for retrieving information about stylesheet properties: `getOutputProperties`, which returns a list of properties as a `java.util.Properties` object, and `getOutputProperty`, which returns a property value.

The TransformerFactory class

If you want an Avaki view to use a custom transformation, the class you specify for the view's style sheet must be an extension of `AbstractTransformerFactory`.

When you extend `AbstractTransformerFactory`, your subclass must include a method like this:

```
public Templates newTemplates(Source source)
    throws TransformerConfigurationException
{
    // TO DO:
    // return an instance of your implementation of
    // Templates here
}
```

The Templates class

When you implement `Templates`, you need a method like this:

```
public synchronized Transformer newTransformer()
    throws TransformerConfigurationException
{
    // TO DO:
    // return an instance of your AbstractTransformer
    // subclass here
}
```

Once you have implemented these classes, the grid server instantiates your transformer using the following algorithm:

- First the grid server reflectively instantiates the transformer factory from the supplied class name. Note that your Factory class must have a no argument constructor.
- Next, the grid server calls `factory.newTemplates(Source)`, passing in a `StreamSource` to the `styleSheet`.
- The grid server calls the `templates.getTransformer()` to get an instance of the transformer.

- Finally, the grid server calls `transformer.transform(Source, Result)` with a reference using a `StreamSource` to the input document and `StreamResult` to the output document.

Installing your Java transformer

To install your Java transformer take the following steps:

- Step 1** Create a jar file with your transformer.
- Step 2** Make sure you have write permission to the install directory.
- Step 3** Place your jar in

```
<Avaki-install-dir>/jboss/server/grid-server/deploy
    /<mytransformer.jar>
```

The jar hot-deploys—that is, the grid server immediately detects and loads the new jar.

Using your transformer

Once you have installed your transformer, you will use it by providing the name of the transformer factory class (your extension of `AbstractTransformerFactory`) to an Avaki view generator whose generated view you wish to transform. For more about views, see the *Sybase Avaki EII Provisioning and Advanced Data Integration Guide*.

Referring to other documents in your transformer

In addition to the style sheet and the input document, a transformer may refer to additional sources through the `URIResolver` interface. The grid server will set a `URIResolver` on the `TransformerFactory` and the `Transformer` that can be used to get references to other documents in the data grid. It will resolve absolute file paths and URIs starting with the `avaki` scheme. In addition you can resolve relative URIs by using the base argument. For example, all of the following will return a `StreamSource` to the grid file `/Shares/xml/colors.xml`.

```
uriResolver.resolve
    ("avaki:///Shares/xml/colors.xml", null);
uriResolver.resolve
    ("colors.xml", "avaki:///Shares/xml/fabrics.xml");
uriResolver.resolve("/Shares/xml/colors.xml", null);
uriResolver.resolve("colors.xml"
    "/Shares/xml/fabrics.xml");
```

Logging errors

The grid server sets an `ErrorListener` on the `TransformerFactory` and the `Transformer`. This object can be used to report errors that will be logged to the grid server log. (The grid server log is in `<avaki-install-directory>/jboss/server/grid-server/log/server.log`.)

The `ErrorListener` interface provides three logging levels: `warn`, `error`, and `fatal`. These levels correspond to the grid server logging levels `WARN`, `ERROR`, and `FATAL`. Errors are logged to the category `io.view.update.stylesheet`.

For more on logging, see the *Sybase Avaki EII Administration Guide*.

Advanced database management

This appendix describes advanced settings that you can configure for database connectors and database operations. These sections explain how to create attributes that specify advanced settings:

- [“Configuring the JDBC driver JAR file path,”](#) below
- [“Restricting database operation output”](#) on page 249
- [“Configuring batch mode settings”](#) on page 250
- [“Configuring SQL calls”](#) on page 251
- [“Configuring database operation timeouts”](#) on page 253
- [“Configuring database operation fetch size”](#) on page 254

For detailed information about creating attributes, see the *Sybase Avaki EII Administration Guide*.

Configuring the JDBC driver JAR file path

If you need two versions of the same JDBC driver on a grid server, you must create an attribute that specifies the path to a JDBC driver JAR file that the database connector will use to get the driver classes. Do *not* put the second JDBC driver’s JAR file in the <Avaki-install-dir>/drivers directory. Instead, put the JAR file in any location on the file system that is local to the grid server.

To create an attribute that specifies a path to an additional JDBC driver JAR file, do the following:

Step 1 Navigate to the View Database Connectors screen:

Home > Database provisioning > Manage database connectors

View Database Connectors

Grid domain:

You are viewing the database connectors in the current domain.

Name	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security
MyDBConnector	View/Edit	Test	Operations	SQL Views	Remove	Attributes	Security

The system displays a list of the database connectors in the current grid domain.

Step 2 Click the **Attributes** link to the right of the database connector for which you want to create a new attribute. The system displays the attributes for the database connector.

User defined Attributes

A user-defined attribute may be one of the following types:

Type	Description	Format
String	Text	Any characters
Integer	Any whole quantity	Any integer
Float	A numeric value that can be fractional or very large	Any numeric value
Date	Year, month, and date	yyyy-mm-dd
Time	Hour, minute and second that an event occurs	hh:mm:ss
Timestamp	A precise time and date	yyyy-mm-dd hh:mm:ss.fffff

This item currently has no user attributes.

To add a new attribute, specify a name, type, and value in the fields below:

Name: Type: Value:

Step 3 Create the following attribute:

- **Name:** dbconn/DriverJarPath
- **Type:** String
- **Value:** Specify the local path to the JDBC driver JAR file.

Restricting database operation output

You can restrict the maximum number of rows that a database operation will return from a back-end database.

To restrict a database operation's row output, create an attribute that specifies the maximum number of rows returned, as follows:

Step 1 Navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations

Grid domain:

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

Step 2 Click the **Attributes** link to the right of the database operation for which you want to create a new attribute. The system displays the attributes for the database operation.

User-defined Attributes

A user-defined attribute may be one of the following types:

Type	Description	Format
String	Text	Any characters
Integer	Any whole quantity	Any integer
Float	A numeric value that can be fractional or very large	Any numeric value
Date	Year, month, and date	yyyy-mm-dd
Time	Hour, minute and second that an event occurs	hh:mm:ss
Timestamp	A precise time and date	yyyy-mm-dd hh:mm:ss.ffffff

This item currently has no user attributes.

To add a new attribute, specify a name, type, and value in the fields below:

Name: Type: Value:

Step 3 Create the following attribute:

- **Name:** dbop/MaxRowsReturned

- **Type:** Integer
- **Value:** Specify a positive integer to configure the maximum number of rows that the database operation will return.

Configuring batch mode settings

If a database operation is configured to perform bulk batch updates from XML or result set input, you can create an attribute that sets the update batch size or specifies the number of individual updates to be added before the JDBC driver calls the `executeBatch()` method to send out any batched-up callable statements (see the *Sybase Avaki EII API Guide* for details about the `executeBatch()` method). Each time the specified number of updates are processed, the JDBC `executeBatch()` method will be invoked. The default is to process all the updates in a single large batch.

To create an attribute that specifies the batch update size for a database operation, do the following:

Step 1 Navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations

Grid domain:

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

- Step 2** Click the **Attributes** link to the right of the database operation for which you want to create a new attribute. The system displays the attributes for the database operation.

User-defined Attributes

A user-defined attribute may be one of the following types:

Type	Description	Format
String	Text	Any characters
Integer	Any whole quantity	Any integer
Float	A numeric value that can be fractional or very large	Any numeric value
Date	Year, month, and date	yyyy-mm-dd
Time	Hour, minute and second that an event occurs	hh:mm:ss
Timestamp	A precise time and date	yyyy-mm-dd hh:mm:ss.ffffff

This item currently has no user attributes.

To add a new attribute, specify a name, type, and value in the fields below:

Name: Type: Value:

- Step 3** Create the following attribute:

- **Name:** dbop/BatchSize
- **Type:** Integer
- **Value:** Specify a positive integer to configure the update batch size or the number of individual updates to be added before calling the `executeBatch()` method.

Configuring SQL calls

By default, Avaki uses the `JDBC Statement.execute()` method to run SQL through a JDBC driver. If the JDBC driver does not support this method or if it is not efficient to call this method, you can configure Avaki to call one of the following methods:

- `Statement.executeQuery()`: Call this method if your SQL performs a query, and set the attribute to “query”.
- `Statement.executeUpdate()`: Call this method if your SQL performs an update, and set the attribute to “update”.

To create an attribute that configures Avaki to call `Statement.executeQuery()` or `Statement.executeUpdate()`, do the following:

Step 1 Navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations

Grid domain:

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

Step 2 Click the **Attributes** link to the right of the database operation for which you want to create a new attribute. The system displays the attributes for the database operation.

User-defined Attributes

A user-defined attribute may be one of the following types:

Type	Description	Format
String	Text	Any characters
Integer	Any whole quantity	Any integer
Float	A numeric value that can be fractional or very large	Any numeric value
Date	Year, month, and date	yyyy-mm-dd
Time	Hour, minute and second that an event occurs	hh:mm:ss
Timestamp	A precise time and date	yyyy-mm-dd hh:mm:ss.ffffff

This item currently has no user attributes.

To add a new attribute, specify a name, type, and value in the fields below:

Name: Type: Value:

Step 3 Create the following attribute:

- **Name:** dbop/ExecutionType
- **Type:** String
- **Value:** Specify one of the following values:

Value	Description
query	Call <code>Statement.executeQuery()</code> if your SQL performs a query.
update	Call <code>Statement.executeUpdate()</code> if your SQL performs an update.
generic	Call <code>Statement.execute()</code> , the default method.

Configuring database operation timeouts

You can configure a value that specifies how much time (in seconds) a database should spend trying to execute a database operation. If the database operation is not executed during that period, the execution attempt is aborted. If you set a value for this attribute, the database operation calls the `Statement.setQueryTimeout()` method with the specified value before executing any SQL statements. For detailed information about `Statement.setQueryTimeout()`, see the Java API documentation or the documentation for your JDBC driver.

To create an attribute that specifies the database operation timeout settings, do the following:

- Step 1** Navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations

Grid domain:

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

- Step 2** Click the **Attributes** link to the right of the database operation for which you want to create a new attribute. The system displays the attributes for the database operation.

User-defined Attributes

A user-defined attribute may be one of the following types:

Type	Description	Format
String	Text	Any characters
Integer	Any whole quantity	Any integer
Float	A numeric value that can be fractional or very large	Any numeric value
Date	Year, month, and date	yyyy-mm-dd
Time	Hour, minute and second that an event occurs	hh:mm:ss
Timestamp	A precise time and date	yyyy-mm-dd hh:mm:ss.ffffff

This item currently has no user attributes.

To add a new attribute, specify a name, type, and value in the fields below:

Name: Type: Value:

- Step 3** Create the following attribute:
- **Name:** dbop/BackendQueryTimeout
 - **Type:** Integer
 - **Value:** Specify an integer for the maximum time (in seconds) before the attempt to execute the database operation is terminated.

Configuring database operation fetch size

To improve performance, JDBC drivers generally buffer the results they return to applications that invoke database operations on them. The application processes one row at a time, but when the driver gets rows from the database, it gets several at once. The number of rows retrieved is called the fetch size. In most circumstances, the driver's default fetch size will be optimal; you also have the ability to configure a database connector to use a different default for all of its database operations (see the "JDBC fetch size" bullet item on [page 5](#)). But if you want a particular database operation to use a non-default fetch size, you can configure it accordingly.

To create an attribute that specifies the fetch size for a particular database operation, do the following:

Step 1 Navigate to the View Database Operations screen:

Home > Database provisioning > Manage database operations

View Database Operations

Grid domain:

You are viewing the database operations in the current domain.

Sort by Name Connector

Name	Connector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata
MyDBOperation	MyDBConnector	View/Edit	Execute	Schedule	Remove	Attributes	Security	Metadata

Step 2 Click the **Attributes** link to the right of the database operation for which you want to create a new attribute. The system displays the attributes for the database operation.

User-defined Attributes

A user-defined attribute may be one of the following types:

Type	Description	Format
String	Text	Any characters
Integer	Any whole quantity	Any integer
Float	A numeric value that can be fractional or very large	Any numeric value
Date	Year, month, and date	yyyy-mm-dd
Time	Hour, minute and second that an event occurs	hh:mm:ss
Timestamp	A precise time and date	yyyy-mm-dd hh:mm:ss.ffffff

This item currently has no user attributes.

To add a new attribute, specify a name, type, and value in the fields below:

Name: Type: Value:

Step 3 Create the following attribute:

- **Name:** dbop/FetchSize
- **Type:** Integer
- **Value:** Specify an integer for the number of rows to be used by the driver as the buffer size for this database operation.

Data service schema

This appendix describes the XML schema that specifies the contents of an Avaki data service. For details about creating data services, see [Chapter 2, “Basic data integration”](#).

The data service description schema is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 3 U
(http://www.xmlspy.com) by William Tam (Avaki Corp) -->
<!--W3C Schema generated by XMLSPY v2004 rel. 3 U
(http://www.xmlspy.com) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="class" type="xs:string"/>
  <xs:element name="dataService">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="description"/>
        <xs:element ref="inputParameter" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element ref="inputSource" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element ref="outputStream" minOccurs="0"/>
        <xs:element ref="urlLogicBox" minOccurs="0"/>
        <xs:element ref="logicBox" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="description" type="xs:string"/>
  <xs:element name="inputParameter">
    <xs:complexType>
```

```

    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="description" minOccurs="0"/>
      <xs:element ref="type" minOccurs="0"/>
      <xs:element ref="isList" minOccurs="0"/>
      <xs:element ref="ref" minOccurs="0"/>
      <xs:element ref="value" minOccurs="0"/>
      <xs:element ref="values" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="inputSource">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="description"/>
      <xs:element ref="type"/>
      <xs:element ref="inputStream"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="inputStream">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="class" minOccurs="0"/>
      <xs:element ref="target" minOccurs="0"/>
      <xs:element ref="ref" minOccurs="0"/>
      <xs:element ref="inputParameter" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="isList" type="xs:boolean"/>
<xs:element name="jarurl" type="xs:string"/>
<xs:element name="name" type="xs:string"/>
<xs:element name="outputStream">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="description" minOccurs="0"/>
      <xs:element ref="type" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ref" type="xs:string"/>
<xs:element name="target">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ref" minOccurs="0"/>
      <xs:element ref="value" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

```

```

    </xs:complexType>
  </xs:element>
  <xs:element name="type" type="xs:string"/>
  <xs:element name="value" type="xs:string"/>
  <xs:element name="logicBox">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="class"/>
        <xs:element ref="initParameter" minOccurs="0"
          maxOccurs="unbounded"/>
        <xs:element ref="inputStream" minOccurs="0"
          maxOccurs="unbounded"/>
        <xs:element ref="outputStream" minOccurs="0"
          maxOccurs="unbounded"/>
        <xs:element ref="inputParameter" minOccurs="0"
          maxOccurs="unbounded"/>
        <xs:element ref="coherenceWindow" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="urlLogicBox">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="jarurl"/>
        <xs:element ref="inputStream" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element ref="outputStream" minOccurs="0"/>
        <xs:element ref="inputParameter" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element ref="initParameter" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element ref="coherenceWindow" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="initParameter">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="value" minOccurs="0"/>
        <xs:element ref="values" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="values">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="value" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```
    <xs:element name="coherenceWindow" type="xs:integer"/>  
</xs:schema>
```

The following sections describe the components of the schema.

Elements:

- class
- coherenceWindow
- dataService
- description
- initParameter
- inputParameter
- inputSource
- inputStream
- isList
- jarurl
- logicBox
- name
- outputStream
- ref
- target
- type
- urlLogicBox
- value
- values

Element: class



Type: xs:string

Used by: elements inputStream, logicBox

Source: <xs:element name="class" type="xs:string"/>

Element: coherenceWindow

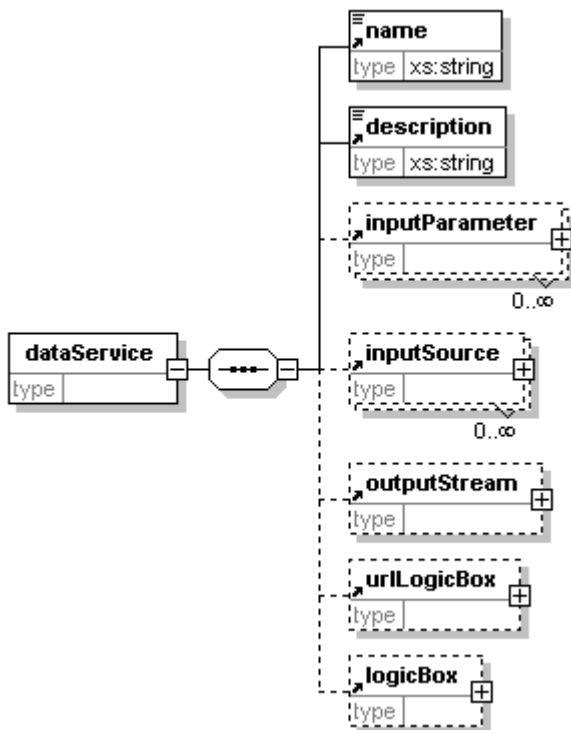


Type: xs:integer

Used by: elements logicBox, urlLogicBox

Source: <xs:element name="coherenceWindow" type="xs:integer"/>

Element: dataService



Children: name, description, inputParameter, inputSource, outputStream, urlLogicBox, logicBox

Source:

```

<xs:element name="dataService">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="description"/>
      <xs:element ref="inputParameter" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref="inputSource" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref="outputStream" minOccurs="0"/>
      <xs:element ref="urlLogicBox" minOccurs="0"/>
      <xs:element ref="logicBox" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
  
```

Element: description

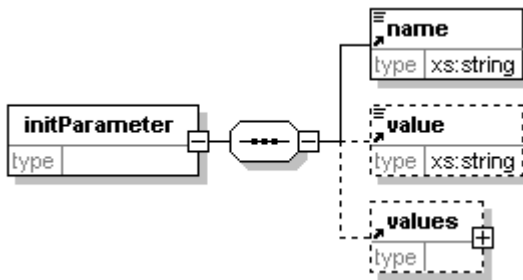


Type: xs:string

Used by: elements dataService, inputParameter, inputSource, outputStream

Source: <xs:element name="description" type="xs:string"/>

Element: initParameter



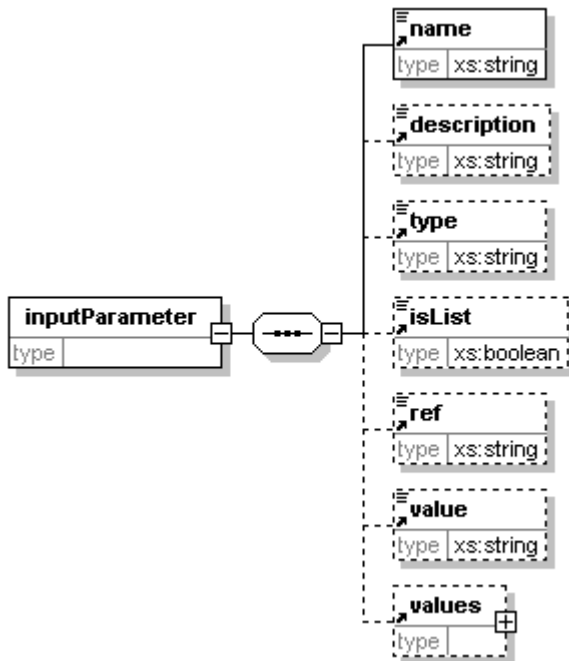
Children: name, value, values

Used by: elements logicBox, urlLogicBox

Source:

```
<xs:element name="initParameter">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="value" minOccurs="0"/>
      <xs:element ref="values" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element: inputParameter



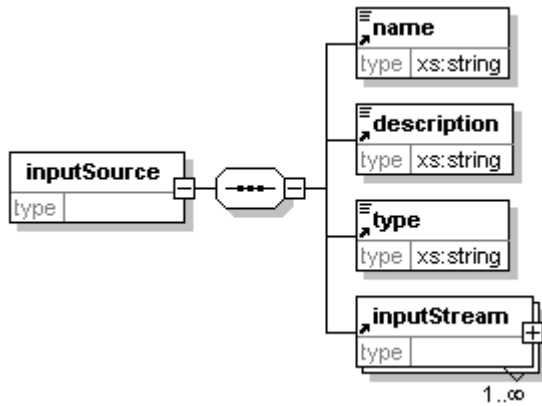
Children: name, description, type, isList, ref, value, values

Used by: elements dataService, inputStream, logicBox, urlLogicBox

Source:

```
<xs:element name="inputParameter">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="description" minOccurs="0"/>
      <xs:element ref="type" minOccurs="0"/>
      <xs:element ref="isList" minOccurs="0"/>
      <xs:element ref="ref" minOccurs="0"/>
      <xs:element ref="value" minOccurs="0"/>
      <xs:element ref="values" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element: inputSource



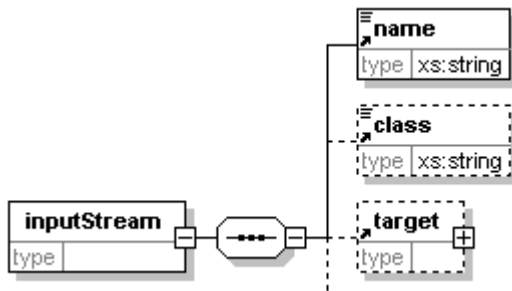
Children: name, description, type, inputStream

Used by: element dataService

Source:

```
<xs:element name="inputSource">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="description"/>
      <xs:element ref="type"/>
      <xs:element ref="inputStream"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element: inputStream



Children: name, class, target, ref, inputParameter

Used by: elements inputSource, logicBox, urlLogicBox

Source:

```
<xs:element name="inputStream">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="class" minOccurs="0"/>
      <xs:element ref="target" minOccurs="0"/>
      <xs:element ref="ref" minOccurs="0"/>
      <xs:element ref="inputParameter" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element: isList



Type: xs:boolean

Used by: element inputParameter

Source: `<xs:element name="isList" type="xs:boolean"/>`

Element: jarurl

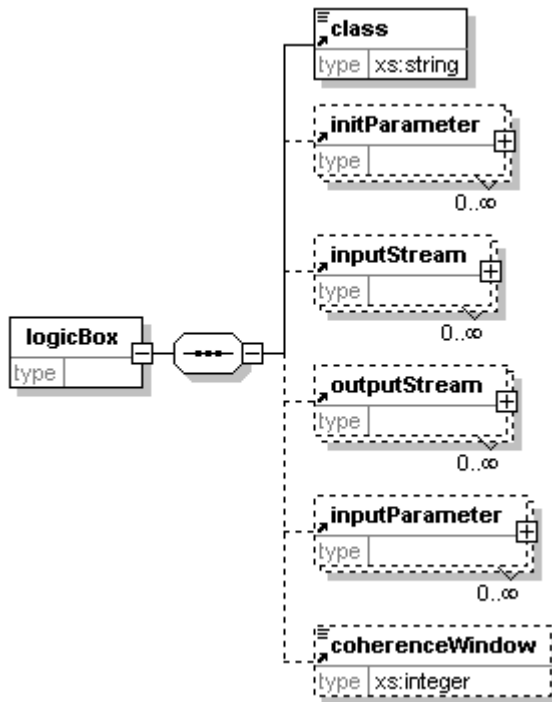


Type: xs:string

Used by: element urlLogicBox

Source: <xs:element name="jarurl" type="xs:string"/>

Element: logicBox



Children: class, initParameter, inputStream, outputStream, inputParameter, coherenceWindow

Used by: element dataService

Source:

```
<xs:element name="logicBox">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="class"/>
      <xs:element ref="initParameter" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref="inputStream" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref="outputStream" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref="inputParameter" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref="coherenceWindow" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element: name

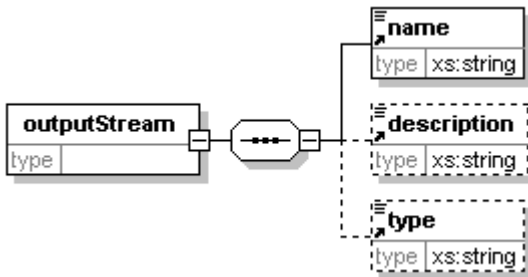


Type: xs:string

Used by: elements dataService, initParameter, inputParameter, inputSource, inputStream, outputStream

Source: `<xs:element name="name" type="xs:string"/>`

Element: outputStream



Children: name, description, type

Used by: elements dataService, logicBox, urlLogicBox

Source:

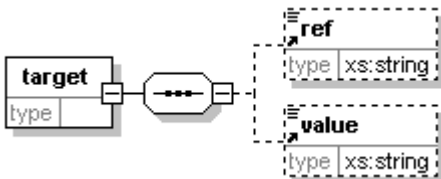
```
<xs:element name="outputStream">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="description" minOccurs="0"/>
      <xs:element ref="type" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element: ref

Type: xs:string

Used by: elements inputParameter, inputStream, target

Source: `<xs:element name="ref" type="xs:string"/>`

Element: target

Children: ref, value

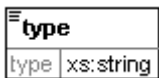
Used by: element inputStream

Source:

```

<xs:element name="target">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ref" minOccurs="0"/>
      <xs:element ref="value" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

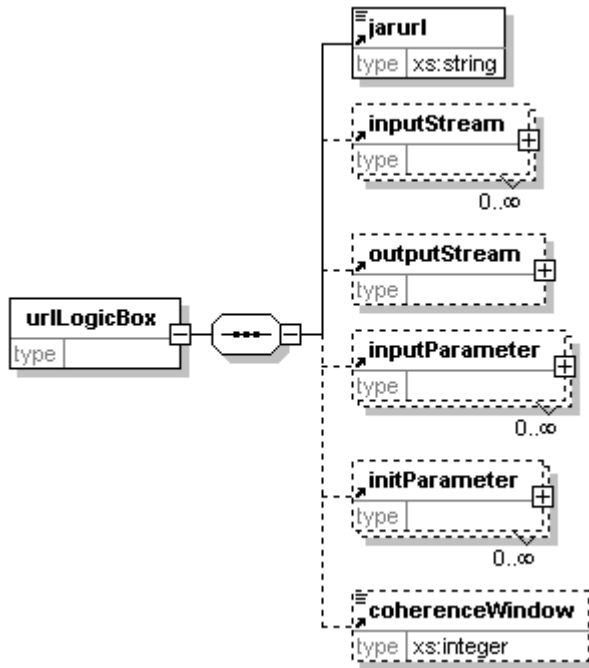
Element: type

Type: xs:string

Used by: elements inputParameter inputSource outputStream

Source: `<xs:element name="type" type="xs:string"/>`

Element: urlLogicBox



Children: jarurl, inputStream, outputStream, inputParameter, initParameter, coherenceWindow

Used by: element dataService

Source:

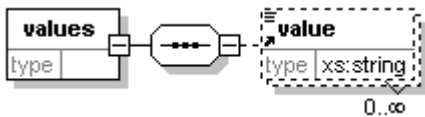
```
<xs:element name="urlLogicBox">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="jarurl"/>
      <xs:element ref="inputStream" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref="outputStream" minOccurs="0"/>
      <xs:element ref="inputParameter" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref="initParameter" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref="coherenceWindow" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element: value

Type: xs:string

Used by: elements initParameter, inputParameter, target, values

Source: <xs:element name="value" type="xs:string"/>

Element: values

Children: value

Used by: elements initParameter, inputParameter

Source:

```

<xs:element name="values">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="value" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Data representation in Avaki

Avaki software has several flexible capabilities, such as data services and view generators, that allow users to manipulate data in any format that is appropriate for their application. However, Avaki uses SQL rowsets and XML as the primary means for representing data.

A rowset is a self-describing sequence of rows. Each row consists of several named and typed columns. Specifically, a rowset is an implementation of either the `java.sql.RowSet` interface or the `java.sql.ResultSet` interface. Avaki uses implementations of both interfaces for data transfer and data manipulation. These implementations are available via published interfaces—either the JDBC driver or the interfaces available to the developer of a data service.

If you are accessing Avaki via an ODBC/JDBC bridge, you will work with the corresponding ODBC abstractions.

Avaki also includes support for representing and manipulating data using XML. In data services and view generators, for example, you can use XSLT to perform data integration and transformation operations.

A unique capability of Avaki is the ability to transform rowsets into XML on demand and to use either rowsets or XML as batch input to database operations. This conversion is on the fly; the XML is produced from the rowset as needed. Thus, a large rowset rendered in XML does not need to be represented in memory all at once, either as a rowset or as XML. As a result, Avaki dovetails well with streaming approaches to XML processing (such as those built around SAX parsing). Of course, if your approach

to XML processing involves representing the document in memory all at once (building a DOM, for instance), you can do that too.

The sections that follow describe how rowsets and XML can be used in a data grid.

Rowset objects

You can use the following objects to produce rowsets:

Object	Description
Database operations	<p>When you execute a database operation, the rowset returned from the back-end database is transformed into an Avaki implementation, routed through the local cache service, and made available to the caller. In this case, the Avaki rowset implementation that is used does not require that the whole rowset be resident in memory, which allows for very large rowsets to be returned.</p> <p>The rowset is streamed back to the program that invoked the database operation, which could be a client application running the Avaki JDBC driver, a data service, or a view generator. Results start streaming back to the caller as soon as the database operation reads the first row produced by the back-end database.</p>
Data services	<p>If you create a data service in Avaki Studio, a rowset is transparently created for you; it represents the Result operator. If you write your own data service plug-in in Java, you can use the <code>com.avaki.core.services.dataservice.api.RowSetFactory</code> class to create rowsets. Data services that produce rowsets can be used as input to other data services that expect rowsets as input.</p> <p>Data services can also produce and consume XML. See below.</p>

Rowsets and XML as inputs

You can use the following objects as inputs:

Object	Description
Data services	<p>A data service can use a database operation, other data service, or any grid file as an input source. If you write your own data service plug-in in Java, the plug-in can handle rowset input (from database operations and data services) in two ways:</p> <ul style="list-style-type: none"> • As rowsets, by calling <code>InputSource.getResultSet()</code>, or • As XML, by calling <code>InputSources.getInputStream()</code>. <p>In the latter case, the returned stream object will transform the rows in the rowset to their corresponding XML representation (see Appendix D, “Avaki rowset XML”) as bytes are read from the stream.</p>
View generators	<p>View generators always treat input as XML or as a raw stream. If the input comes into the view generator as a rowset, it is transformed into the corresponding XML on the fly.</p>

Usage scenarios

Here are some examples of ways you can use data services and view generators:

- You have a file in CSV format that contains information that needs to be joined with data from a relational database. One approach is to create a data service that uses a plug-in that converts a CSV file into a result set. The plug-in reads the CSV file that has been made available via an Avaki share and produces a rowset. You then use Avaki Studio to create a database operation to encapsulate the query that will retrieve the necessary data from your relational database and combine the two rowsets. For more information on how this plug-in works, see `<avaki-install-dir>/examples/plugins/CSVToResultSet/readme.txt`. For information about using Avaki Studio to create database operations and join rowsets, see *Data Integration with Sybase Avaki Studio*.
- You have to generate a CSV file that contains the results of a data service or database operation. To do this, you can use an Avaki view generator that takes the data

service or database operation as input and uses the built-in CSV transformation capability.

- You need to move some data from a database operation or from a data service that produces rowsets (or XML) into another relational store. To do this, create a database operation that performs the insert or update function, and then create a data service that takes the source data (database operation or data service) and the update database operation as input sources. Within the data service plug-in, create a streaming rowset (using `RowSetFactory.createStreamingRowSet(int [] sqlTypes)`). You will then need to downcast the `InputSource` object for the update database operation to `DbopInputSource` and invoke the `execute()` method passing the rowset as a parameter.

Avaki rowset XML

Avaki converts rowsets into an XML representation that contains all the rowset metadata as well as the row and column data. The schema for this representation has two parts as well: the core schema, which is the same for all rowsets, and the rowset-specific schema, in which the element names correspond to the actual column names in the rowset.

In this appendix:

- [“Core schema,”](#) below
- [“Rowset-specific schema”](#) on page 280
- [“Sample XML schema for a database operation”](#) on page 280

Core schema

The core schema defines the overall structure of the XML document plus the structure of the metadata. A portion of a sample document is included below.

```
<?xml version="1.0" encoding="UTF-8" ?>
<results
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <rowset>
    <metadata>
      <column-count>9</column-count>
      <column-definition>
```

```

    <column-index>1</column-index>
    <class-name>java.lang.String</class-name>
    <column-display-size>16</column-display-size>
    <column-label>SIC</column-label>
    <column-name>SIC</column-name>
    <column-type>VARCHAR</column-type>
    <column-type-name>VARCHAR</column-type-name>
    <column-precision>0</column-precision>
    <column-scale>0</column-scale>
    <auto-increment>false</auto-increment>
    <case-sensitive>false</case-sensitive>
    <currency>false</currency>
    <nullable>columnNullable</nullable>
    <read-only>false</read-only>
    <signed>false</signed>
    <searchable>true</searchable>
  </column-definition>
  ...
</metadata>
<data>
  <row>

```

This is where result-set-specific elements are found.

```

    </row>
    ...
  </data>
</rowset>

```

The core schema allows for multiple rowsets to be returned.

```

  <update-count> </update-count>
</results>

```

The root element is `<results>`. Within `<results>`, there can be any number of `<rowset>` elements, followed by a single `<update-count>` element. The update count is relevant for database operations that perform insert and update operations and reflects the number of rows updated by the operation.

The bulk of the document comprises one or more `<rowset>`s. A rowset comprises two subelements: `<metadata>` and `<data>`.

The structure of `<metadata>` is as follows:

- A single `<column-count>` element that contains the number of columns in the rowset

- One or more <column-definition> elements (the number corresponding to <column-count>) that describe the individual columns in each row. Each column definition contains the following elements:

Element	Description
<column-index>	The ordinal position of the column in the row
<class-name>	The java class that you should use when retrieving this column via JDBC.
<column-display-size>	The column's maximal width in characters
<column-type>	The SQL type of the column
<column-type-name>	The SQL type name of the column
<column-precision>	The column's number of decimal digits
<column-scale>	The number of digits to the right of the decimal point
<auto-increment>	Indicates whether the column is automatically numbered and therefore read-only (true/false)
<case-sensitive>	Indicates whether or not the column is case sensitive (true/false)
<currency>	Indicates whether or not the column represents a currency value (true/false)
<nullable>	Indicates whether or not the column allows nulls. Values are one of columnNoNulls, columnNullable, or columnNullableUnknown
<read-only>	Indicates whether the column is read-only (true/false)
<signed>	Indicates whether column values are signed numbers (true/false)
<searchable>	Indicates whether the column can be used in a where clause (true/false)

The structure of the <data> element is simple. It consists of one or more <row> elements. The row element is where the generic schema ends and the rowset-specific schema begins.

Rowset-specific schema

The rowset-specific schema describes the elements within a row. Each column is represented by an element that has the same name as the column. If the value of the column is null, then the attribute “xsi:nil” is true for that element.

Here is an example row:

```
<row>
  <SIC>3441</SIC>
  <COMPANY>Nisshin Steel</COMPANY>
  <ADDRESS1>Shinkokusai Building</ADDRESS1>
  <ADDRESS2>-1, Marunouchi 3-chome</ADDRESS2>
  <CITY>Tokyo</CITY>
  <STATE xsi:nil="true" />
  <ZIP>100-8366</ZIP>
  <COUNTRY>JAPAN</COUNTRY>
  <ID>109</ID>
</row>
```

Sample XML schema for a database operation

This section gives the XML schema generated by Avaki for a database operation whose SQL statement is:

```
SELECT SIC,COMPANY, ADDRESS1, ADDRESS2, CITY, STATE, ZIP,
       COUNTRY, ID
from demo.CUSTOMER_SOR
```

Note that the generated schema includes, as annotations, a lot of information about the database operation itself. The idea is that XML schema is used as a convenient, standard grammar for describing the “shape” of data regardless of whether it is used and manipulated as XML or as native sets.

Sample XML schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- GENERAL DEFINITIONS -->
  <xs:simpleType name="update-countType">
    <xs:restriction base="xs:string"/>
  </xs:simpleType>

  <xs:simpleType name="column-countType">
```



```
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="column-indexType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="class-nameType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="column-display-sizeType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="column-labelType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="column-nameType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="column-typeType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="column-type-nameType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="column-precisionType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="column-scaleType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="auto-incrementType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="case-sensitiveType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="currencyType">
```

```

    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="nullableType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="read-onlyType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="signedType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="searchableType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:complexType name="column-definitionType">
    <xs:sequence>
        <xs:element name="column-index" type="column-indexType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="class-name" type="class-nameType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="column-display-size" type="column-display-sizeType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="column-label" type="column-labelType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="column-name" type="column-nameType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="column-type" type="column-typeType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="column-type-name" type="column-type-nameType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="column-precision" type="column-precisionType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="column-scale" type="column-scaleType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="auto-increment" type="auto-incrementType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="case-sensitive" type="case-sensitiveType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="currency" type="currencyType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="nullable" type="nullableType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="read-only" type="read-onlyType"

```

```

        minOccurs="0" maxOccurs="1"/>
    <xs:element name="signed" type="signedType"
        minOccurs="0" maxOccurs="1"/>
    <xs:element name="searchable" type="searchableType"
        minOccurs="0" maxOccurs="1"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="dataType">
    <xs:sequence>
        <xs:element name="row" type="rowType"
            minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="metadataType">
    <xs:sequence>
        <xs:element name="column-count" type="column-countType"/>
        <xs:element name="column-definition" type="column-definitionType"
            minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="rowsetType">
    <xs:sequence>
        <xs:element name="metadata" type="metadataType"
            minOccurs="0" maxOccurs="1"/>
        <xs:element name="data" type="dataType"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="resultsType">
    <xs:sequence>
        <xs:element name="rowset" type="rowsetType"/>
        <xs:element name="update-count" type="update-countType" />
    </xs:sequence>
</xs:complexType>

<xs:element name="results" type="resultsType"/>

<!-- DBOP SPECIFIC DEFINITIONS -->

<xs:annotation>
    <xs:appinfo>
        <qualifiedDBOPName>mydom.CUSTOMER_SOR.All_Customers</qualifiedDBOPName>
        <SQLStatement>SELECT SIC,COMPANY, ADDRESS1, ADDRESS2, CITY, STATE, ZIP,
COUNTRY, ID from demo.CUSTOMER_SOR</SQLStatement>
        <parameters>

```

```

    </parameters>
  </xs:appinfo>
</xs:annotation>

<xs:complexType name="rowType">
  <xs:sequence>
    <xs:element name="SIC" type="xs:string" nillable="true">
      <xs:annotation>
        <xs:appinfo>
          <column-index>1</column-index>
          <class-name>java.lang.String</class-name>
          <column-display-size>16</column-display-size>
          <column-label>SIC</column-label>
          <column-name>SIC</column-name>
          <column-type>VARCHAR</column-type>
          <column-type-name>VARCHAR</column-type-name>
          <column-precision>0</column-precision>
          <column-scale>0</column-scale>
          <auto-increment>false</auto-increment>
          <case-sensitive>false</case-sensitive>
          <currency>false</currency>
          <nullable>columnNullable</nullable>
          <read-only>false</read-only>
          <signed>false</signed>
          <searchable>true</searchable>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
    <xs:element name="COMPANY" type="xs:string" nillable="true">
      <xs:annotation>
        <xs:appinfo>
          <column-index>2</column-index>
          <class-name>java.lang.String</class-name>
          <column-display-size>64</column-display-size>
          <column-label>COMPANY</column-label>
          <column-name>COMPANY</column-name>
          <column-type>VARCHAR</column-type>
          <column-type-name>VARCHAR</column-type-name>
          <column-precision>0</column-precision>
          <column-scale>0</column-scale>
          <auto-increment>false</auto-increment>
          <case-sensitive>false</case-sensitive>
          <currency>false</currency>
          <nullable>columnNullable</nullable>
          <read-only>false</read-only>
          <signed>false</signed>
          <searchable>true</searchable>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>

```

```

    </xs:annotation>
</xs:element>
<xs:element name="ADDRESS1" type="xs:string" nillable="true">
  <xs:annotation>
    <xs:appinfo>
      <column-index>3</column-index>
      <class-name>java.lang.String</class-name>
      <column-display-size>64</column-display-size>
      <column-label>ADDRESS1</column-label>
      <column-name>ADDRESS1</column-name>
      <column-type>VARCHAR</column-type>
      <column-type-name>VARCHAR</column-type-name>
      <column-precision>0</column-precision>
      <column-scale>0</column-scale>
      <auto-increment>false</auto-increment>
      <case-sensitive>false</case-sensitive>
      <currency>false</currency>
      <nullable>columnNullable</nullable>
      <read-only>false</read-only>
      <signed>false</signed>
      <searchable>true</searchable>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="ADDRESS2" type="xs:string" nillable="true">
  <xs:annotation>
    <xs:appinfo>
      <column-index>4</column-index>
      <class-name>java.lang.String</class-name>
      <column-display-size>64</column-display-size>
      <column-label>ADDRESS2</column-label>
      <column-name>ADDRESS2</column-name>
      <column-type>VARCHAR</column-type>
      <column-type-name>VARCHAR</column-type-name>
      <column-precision>0</column-precision>
      <column-scale>0</column-scale>
      <auto-increment>false</auto-increment>
      <case-sensitive>false</case-sensitive>
      <currency>false</currency>
      <nullable>columnNullable</nullable>
      <read-only>false</read-only>
      <signed>false</signed>
      <searchable>true</searchable>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="CITY" type="xs:string" nillable="true">
  <xs:annotation>

```

```

<xs:appinfo>
  <column-index>5</column-index>
  <class-name>java.lang.String</class-name>
  <column-display-size>64</column-display-size>
  <column-label>CITY</column-label>
  <column-name>CITY</column-name>
  <column-type>VARCHAR</column-type>
  <column-type-name>VARCHAR</column-type-name>
  <column-precision>0</column-precision>
  <column-scale>0</column-scale>
  <auto-increment>false</auto-increment>
  <case-sensitive>false</case-sensitive>
  <currency>false</currency>
  <nullable>columnNullable</nullable>
  <read-only>false</read-only>
  <signed>false</signed>
  <searchable>true</searchable>
</xs:appinfo>
</xs:annotation>
</xs:element>
<xs:element name="STATE" type="xs:string" nillable="true">
  <xs:annotation>
    <xs:appinfo>
      <column-index>6</column-index>
      <class-name>java.lang.String</class-name>
      <column-display-size>2</column-display-size>
      <column-label>STATE</column-label>
      <column-name>STATE</column-name>
      <column-type>CHAR</column-type>
      <column-type-name>CHAR</column-type-name>
      <column-precision>0</column-precision>
      <column-scale>0</column-scale>
      <auto-increment>false</auto-increment>
      <case-sensitive>false</case-sensitive>
      <currency>false</currency>
      <nullable>columnNullable</nullable>
      <read-only>false</read-only>
      <signed>false</signed>
      <searchable>true</searchable>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="ZIP" type="xs:string" nillable="true">
  <xs:annotation>
    <xs:appinfo>
      <column-index>7</column-index>
      <class-name>java.lang.String</class-name>
      <column-display-size>10</column-display-size>

```

```

    <column-label>ZIP</column-label>
    <column-name>ZIP</column-name>
    <column-type>VARCHAR</column-type>
    <column-type-name>VARCHAR</column-type-name>
    <column-precision>0</column-precision>
    <column-scale>0</column-scale>
    <auto-increment>false</auto-increment>
    <case-sensitive>false</case-sensitive>
    <currency>false</currency>
    <nullable>columnNullable</nullable>
    <read-only>false</read-only>
    <signed>false</signed>
    <searchable>true</searchable>
  </xs:appinfo>
</xs:annotation>
</xs:element>
<xs:element name="COUNTRY" type="xs:string" nillable="true">
  <xs:annotation>
    <xs:appinfo>
      <column-index>8</column-index>
      <class-name>java.lang.String</class-name>
      <column-display-size>32</column-display-size>
      <column-label>COUNTRY</column-label>
      <column-name>COUNTRY</column-name>
      <column-type>VARCHAR</column-type>
      <column-type-name>VARCHAR</column-type-name>
      <column-precision>0</column-precision>
      <column-scale>0</column-scale>
      <auto-increment>false</auto-increment>
      <case-sensitive>false</case-sensitive>
      <currency>false</currency>
      <nullable>columnNullable</nullable>
      <read-only>false</read-only>
      <signed>false</signed>
      <searchable>true</searchable>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="ID" type="xs:string" nillable="true">
  <xs:annotation>
    <xs:appinfo>
      <column-index>9</column-index>
      <class-name>java.lang.String</class-name>
      <column-display-size>8</column-display-size>
      <column-label>ID</column-label>
      <column-name>ID</column-name>
      <column-type>VARCHAR</column-type>
      <column-type-name>VARCHAR</column-type-name>

```

```
    <column-precision>0</column-precision>
    <column-scale>0</column-scale>
    <auto-increment>false</auto-increment>
    <case-sensitive>false</case-sensitive>
    <currency>false</currency>
    <nullable>columnNullable</nullable>
    <read-only>false</read-only>
    <signed>false</signed>
    <searchable>true</searchable>
  </xs:appinfo>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

Glossary

Terms printed in *italics* are defined in the glossary.

access control list

(ACL) A list, for a given file, directory, or other Avaki object, of permissions—read, write, execute, delete, and owner—that control which users and groups can view, modify, invoke, and remove the object, and edit the object’s ACL.

ACL

See *access control list*.

ad-hoc query

A mechanism that lets you directly query a database in SQL. The query must run through an existing Avaki *database connector*. You can run an ad-hoc query using either the CLI or a *JDBC driver*. Ad-hoc queries can be thought of as single-use *database operations*.

attribute

A property of an *Avaki directory*, file, *service*, or other object. Each attribute has a name, a type (string, integer, float, date, time, or timestamp) and a value. System attributes are read-only; you can change the values of other attributes. You can also create new attributes and add them to objects as needed.

authentication service

A *service* associated with an *Avaki domain* that authenticates an Avaki user’s identity and provides security credentials each time the user logs in. Avaki can be configured to use third-party directory services as authentication services for login; for user accounts created directly in the Avaki domain, Avaki uses its own default authentication service.

Avaki directory

Avaki software creates a single, unified namespace that is accessible (subject to Avaki *access control lists*) to all users in the *Avaki domain*. The namespace, called the *data catalog*, is arranged as a hierarchy of Avaki directories (folders). The catalog directory structure is stored by the domain's grid servers and its GDC, while the physical files remain in their original locations in your local file systems. When you work with directories, it's important to distinguish between Avaki directories, which are part of the data catalog, and local directories, which reside in your local file system.

Avaki domain

The basic administrative unit of the Avaki EII system. An Avaki domain consists, at a minimum, of one *grid domain controller* and may also include one or more *grid servers*, *share servers*, *proxy servers*, *data grid access servers*, and *command clients*. See also *domain name*.

Avaki group

A set of users who have the same permissions on one or more Avaki objects. You can use the group name in place of a user name when you set permissions or create *access control lists*.

Avaki installation directory

The directory in your local file system where Avaki software is installed. This is not a *data catalog* directory.

Avaki share

(Also shared directory.) A pointer in the Avaki *data catalog* to a directory or file in the underlying local file system. When you browse the data catalog, Avaki shares look like—and can be accessed like—other Avaki directories. Contrast with *CIFS share*.

Avaki server

A *service* that starts, stops, and monitors other Avaki services on a particular computer. Every server is part of an *Avaki domain*. A server is permanently attached to the computer where it is started. There are several types of server: *data grid access servers*, *grid domain controllers*, *grid servers*, *share servers*, and *proxy servers*.

Avaki Studio

A graphical, metadata-based data integration tool that lets you

- Build data flows by dragging and dropping input sources, operators, and output targets. You can deploy your data flows as *Avaki data services*.
- Import or create *metadata models* and apply them to Avaki objects or use them to build new data services.

In addition, you can use Studio to perform provisioning tasks (creating *database connectors*, *database operations*, *virtual database operations*, and *SQL views*), manipulate *categories*, and edit *ad-hoc queries* and *attributes*.

cache service

(Formerly proxy cache service.) A staging service that stores copies of files, *database operation* results, and *data service* results. Caching improves retrieval performance. To ensure that an object is stored in the cache, you can *pin* a file or directory in the data catalog, or schedule a database operation or data service. A cache service can provide remote caching, local caching, or both. The freshness of cached data is controlled by a data expiration interval that determines how long cached data is considered valid and by a cache coherence window that tells the cache service how often to check whether cached data is still valid. If cached data is too old to satisfy a new request (or is not stored in this cache), the cache service does one of the following:

- If the database operation or data service that produced the data is local to this cache service, the cache service triggers execution of the database operation or data service.
- If the database operation or data service that produced the data is remote from this cache service, this cache service requests the data from the data source's local cache service.

A cache service can be associated with a *data grid access server*, a *grid server*, or a local user in a CLI session. See also *local cache*, *remote cache*, *on-demand caching*, and *scheduled caching*.

category

A mechanism for classifying and organizing the contents of the *data catalog*. Like *Avaki directories*, categories serve as containers for objects in the data catalog. Anything in the data catalog—views, data services, shared files, even Avaki directories themselves—can be assigned to a category. Categories are hierarchical, they have attributes, and Avaki *access control lists* regulate access to them.

CIFS client

A machine that mounts files or directories from the Avaki *data catalog* by connecting to a *CIFS share* through an Avaki *data grid access server*. A CIFS client need not have Avaki software installed. (CIFS—Common Internet File System—is a file-sharing protocol based on the file system implemented by Windows.)

CIFS share

A directory or file that has been exported (shared) from the Avaki *data catalog*. A CIFS share can be mapped into a Windows file system like a network drive. When you browse the Windows file system, CIFS shares look like—and can be accessed like—other files and directories. CIFS shares are created through a *data grid access server*. Contrast with *Avaki share*.

client

Avaki supports several types of client: *Avaki Studio*, *CIFS clients*, *command clients*, *JDBC/ODBC clients*, *NFS clients*, *web clients*, and *WS clients*.

command client

A machine that can issue Avaki commands but does not contribute resources to the *Avaki domain*.

connect port

The connect port on a *grid domain controller*, *grid server*, *data grid access server*, *proxy server*, or *share server* accesses the JNDI naming service or RMI registry for the underlying application server. The connect port is one of many ports that a GDC or server uses to communicate with other Avaki objects. You must supply the connect port number of a target grid server or GDC whenever you connect a new object (another server, a copy of Avaki Studio, or a *command client*, for example) to an *Avaki domain*. When you *interconnect* two Avaki domains, you must supply each domain's connect port number to the other one.

data catalog

A hierarchical structure similar to a file system that encompasses all objects in an *Avaki domain*. The data catalog contains *Avaki directories* and files, *Avaki shares*, *Avaki servers*, *SQL views*, *database operations* and *data services*, and other objects.

data grid access server

(DGAS) An *Avaki server* that makes *Avaki directories* and their contents available to *CIFS clients* and *NFS clients*.

data service

An operation that transforms data obtained from sources in the *data catalog*. Input data can come from any number of sources, including:

- other data services
- data catalog files (which can be *generated views*)
- *Avaki database operations* (which in turn extract the data from relational databases)
- HTTP requests
- Web service invocations

You can generate the code that manipulates the data by creating a *view model* in *Avaki Studio*, or by writing a custom *data service plug-in* using Java, JavaScript, or XSLT. Data service output can be in rowset or XML format. Data services are run by the *execution services* on *grid servers*, they can be scheduled, and their results can be cached.

data service plug-in

The logic for a *data service*, written in Java, JavaScript, or XSLT. Data service plug-ins are modular—you can use the same plug-in for multiple data services. *Avaki Studio* creates data services and plug-ins simultaneously, so if you use Avaki Studio to create data services, you don't have to worry about plug-ins. You can also use the Avaki Plug-in Wizard to create data service plug-ins.

database connector

A mechanism that enables one or more *database operations*, *SQL views*, or *ad-hoc queries* to connect to a relational database.

database operation

(DBOP) A mechanism that can

- extract data from a relational database and deliver it on demand to a *view generator* or a *data service*, or
- modify data in a relational database.

A database operation can be a SQL statement or a stored procedure call.

dependency

A relationship in which an Avaki object requires input from other Avaki objects. A *data service* might require input from one or more *database operations* or from other data services. A *view generator* might depend on a database operation for input. A database operation can serve as an input source for one or more data services or view generators. Generated *SQL views* depend on database operations, virtual database operations, or data services. You can use *Avaki Studio*, the web UI, or the CLI to list input and output dependencies for any data service, database operation, or view.

DGAS

See *data grid access server*.

distributed transaction

A set of related operations (typically SQL operations such as SELECT, INSERT, UPDATE, DELETE, and CALL) that

- involve one or more databases, and
- might lead to unwanted results (such as leaving participating databases in an inconsistent state or producing inconsistent reads) if some of the operations complete and others do not, and therefore
- must all be executed at once, as a single transaction.

The individual operations that make up a distributed transaction are performed by *database operations* that use *database connectors* configured with XA-capable *JDBC drivers*; all the database opera-

tions are executed, using the two-phase commit protocol, by a specially configured *data service*. The two-phase commit protocol is designed to ensure that the participating databases will be left in a consistent state—that is, that all the operations in the distributed transaction will be completed, or none of them will.

domain name

A unique alphanumeric identifier for an *Avaki domain*. The domain name is assigned by the Avaki administrator when the Avaki domain is initialized. The domain name has a maximum length of 30 characters.

enterprise information integration

Execution services execute *data services*. There is an execution service on every *grid server*, and you can configure a pool of execution services for load-sharing. When a pool is in place, a data service can be run by any execution service in its grid server's pool.

exclusion

See *schedule exclusion*.

execution service

Execution services execute *data services*. There is an execution service on every *grid server*, and you can configure a pool of execution services for load-sharing. When a pool is in place, a data service can be run by any execution service in its grid server's pool.

failover

The transition of control from a failing or unreachable primary *grid domain controller* to a secondary grid domain controller.

federated data access

A scheme that allows independently controlled elements to be shared into a single namespace. Files, user accounts, and other objects maintain their separate identities and remain under the control of their owners, but—subject to access controls—the objects can be accessed, managed, and viewed as if they were part of a single system.

GDC

See *grid domain controller*.

generated view

A file created by a *view generator*; it may contain data obtained from a *database operation*, a *data service*, a file, or an HTTP source. Like other files, generated views exist in a local file system and are shared into the *data catalog*.

grid

A heterogeneous group of networked resources that appears and functions as one operating environment. A data grid like the Avaki Enterprise Information Integration (EII) system provides secure, shared access to data.

grid directory

See *Avaki directory*.

grid domain

See *Avaki domain*.

grid domain controller

(GDC) The first server in an *Avaki domain* is the grid domain controller. The GDC maintains a portion of the Avaki domain's namespace and provides authentication services. It can also run Avaki commands, share data, and monitor other servers. (That is, the GDC functions as a *grid server*.) If the domain is configured for *failover*, it has both a primary GDC and a secondary GDC; the secondary is updated at regular intervals and takes over management of the domain if the primary fails. Any Avaki shares managed by the primary are read-only on the secondary.

grid server

An *Avaki server* that maintains a portion of the *Avaki domain*'s namespace, runs Avaki services such as shares, execution services, caches, and searches, and allows you to run Avaki's web UI and execute Avaki commands.

group

See *Avaki group*.

hard link

Provides an alternate name for an item in the *data catalog*. Changes to the object's other names have no effect on the hard link: you can move or change a file's original name and the hard link will still know where to find the file. To delete a hard-linked object, you must remove its original name. Contrast with *soft link*.

interconnect

To create a unidirectional link from one *Avaki domain* to another. Interconnecting lets an Avaki domain make its *data catalog* visible to users in another domain (subject to Avaki access controls).

JDBC driver

JDBC (Java Database Connectivity) drivers allows application programmers to access database data shared in the *data catalog*. When a JDBC driver accesses data, it returns a JDBC result set that's immediately available to your program. JDBC drivers can:

- Call any *data service* in the data catalog
- Call any *database operation* in the data catalog
- Perform SQL `select` operations against *SQL views* in the data catalog

Sybase offers three JDBC drivers for use with Avaki EII software:

- The included Avaki JDBC driver
- jConnect, Sybase's standard JDBC driver
- An XA-capable driver for use with *database connectors* that support *distributed transactions*

link

See *hard link* and *soft link*.

local cache

A *cache service* that runs on the same *grid server* as a *database operation* or a *data service* that generates cachable data. The local cache stores results produced by local database operations and data services so they don't have to execute for every new request. See also *remote cache*.

metadata model

A construct in *Avaki Studio* that expresses a schema by defining a set of tables. A table in a metadata model can be mapped (linked) to an Avaki object such as a *data service* or a *database operation*, or to a table in a relational database. The mapping lets you address each mapped object by the name of the corresponding table in the metadata model. You can also derive a *view model* schema from a metadata model. When you do this, you ensure that the results of any data service deployed from the view model will conform to the metadata model's schema.

NFS client

A machine that mounts the Avaki *data catalog* (or a portion of it) as a directory by connecting to an Avaki *data grid access server*. An NFS client need not have Avaki software installed. (NFS—Network File System—lets you add file systems located on a remote computer to the directory structure on your own computer.)

ODBC

ODBC (Open DataBase Connectivity) is an API for databases on Windows. An ODBC driver (such as the the Sybase Organic ODBC driver included with Sybase ASE) allows Avaki to communicate with Windows database applications.

on-demand caching

A scheme by which an object is cached only if it's used—for example, results are cached when a *database operation* or a *data service* is executed, or a file is cached when a user or application reads it. On-demand caching uses a fixed expiration interval to determine data freshness. On-demand caching is suitable for objects that are rarely accessed or that change at irregular intervals. Contrast with *scheduled caching*.

pin

To mark an *Avaki directory* or file for *scheduled caching*. See also *cache service*.

plug-in

See *data service plug-in*.

primary GDC

See *grid domain controller*.

proxy server

An *Avaki server* that allows *Avaki domains* on opposite sides of a firewall or a Network Address Translator (NAT) to communicate with one another.

queries

See *ad-hoc query*.

query engine

An *Avaki service* that executes SQL queries against the *SQL views* (tables) that make up the *Avaki virtual database*. A query engine analyzes queries, pushes as much of the work as possible down to the underlying relational database (if there is one), and performs the remaining operations (such as joins across tables from different databases) itself. There is a query engine on each *grid server*.

remote cache

A *cache service* that runs on a grid server that is remote from an *Avaki service* (a *database operation* or a *data service*) that generates cachable data. The remote cache stores results produced by distant services so the results don't have to be fetched over the network to satisfy every new request. Users and applications that access remote data through the cache may have access to cached copies even when the remote data source is unavailable. See also *local cache*.

scheduled caching

A scheme by which an object is cached according to a schedule that you create. The schedule specifies when the object is first cached and how often (or following what trigger event, such as a change to a file) the cache is refreshed. If the object is a *data service* or a *database operation*, the schedule runs it

to put fresh results in the cache. Scheduled caching, which overrides other types of caching, is suitable for objects that are updated frequently or on a regular basis. Contrast with *on-demand caching*.

schedule exclusion

A named period of time during which scheduled activities can be prevented from running. You can apply an exclusion to as many schedules as you want. Scheduled activities include refreshing *Avaki shares* and imported user accounts, and caching files, directories, and the results of *database operations*, *data services*, and *generated views*.

secondary GDC

See *grid domain controller*.

service

An Avaki object that performs a function in the domain (stores data or authenticates users, for example). Services provided in Avaki software include *Avaki directories*, *Avaki shares*, *Avaki servers*, *authentication services*, *execution services*, and user accounts.

share

A point of connection between the Avaki *data catalog* and a native file system or file system tool. Avaki supports two kinds of shares: *Avaki shares* and *CIFS shares*.

share server

An *Avaki server* whose only task is to manage *Avaki shares*—local directories that are exported (shared) into the *data catalog*. (Grid servers can also manage shares.)

shared directory

See *Avaki share*.

soft link

A pointer to a particular location (name) in the Avaki *data catalog*. If the object at that location is moved, deleted, or renamed, the soft link leads nowhere. Soft links can be created only in the CLI. Contrast with *hard link*.

SQL view

A virtual table—a *data catalog* entry that represents a table in a relational database, a *database operation*, or a *data service*. SQL views can be created in three ways:

- Provisioned directly from a table in an underlying database
- Generated from a database operation or data service

- Mapped from a database table, a database operation, or a data service, using the *Avaki Studio* metadata model editor

Every SQL view is part of the *Avaki virtual database*. SQL views are treated as relational tables by the *Avaki query engine*. SQL view data can be accessed using standard SQL statements by connecting to Avaki with ODBC or JDBC, or via an *Avaki virtual database operation*.

update notification

A message issued when a *generated view* is updated. A view that receives data from another view can be configured to regenerate itself (using the new data) upon receipt of an update notification.

view generator

A mechanism that does one of the following: extracts data from a file or an HTTP source, obtains data from an *Avaki data service*, or uses an *Avaki database operation* to extract data from a relational database. The view generator can display the data, perform an XSLT transform, save the data as a *generated view* file, and/or update a database. Contrast with *data service*.

view model

The graphical representation of a data flow that you can build in *Avaki Studio*. A view model typically includes one or more input sources (such as *database operations* or *data services*), one or more operations to combine or transform the data, and an output target. When you deploy a view model, it becomes an Avaki data service.

virtual database

The set of all *SQL views* in an *Avaki domain*, including those provisioned from external databases and those generated from *data services* and *database operations*. You can execute SQL queries on the SQL views in the virtual database as if they were tables in a single database.

virtual database operation

A *database operation* whose source database is the *Avaki virtual database* itself. Use virtual database operations if you want to encapsulate and reuse SQL SELECT queries against *SQL views* (provisioned or generated).

web services client

See *WS client*.

WS client

(Also web services client.) A tool or a piece of code that is part of a customer application and that makes SOAP calls to web services on an Avaki grid server. The SOAP calls can request data from the *Avaki data catalog*, from a *database operation*, or from a *data service*.

Master Index

In electronic copies of this book, the index links to other books in the documentation set work only as long as the PDF files are stored in the same directory.

Key

AD: *Administration Guide*

API: *API Guide*

C: *Command Reference*

O: *Overture*

P: *Provisioning & Advanced Data Integration Guide*

S: *Data Integration with Avaki Studio*

Symbols

* asterisks in command syntax AD:xvi, C:xv, P:xi
- hyphens in command syntax AD:xvi, C:xvi, P:xii
+ plus signs in command syntax AD:xvi, C:xv, P:xi
.amm files S:11
.avm files S:11
.js files S:11
.jsi files S:11, S:75
 sample S:115
.NET
 AvakiAPI.disco WSDL discovery file API:3
 sample web services client API:9
 SSL certificates API:9
.project files S:11
<> angle brackets in command syntax AD:xvi, API:vii, C:xv,
 P:xi
= equal signs in command syntax AD:xvi, C:xvi, P:xii
[] square brackets in command syntax AD:xv, C:xv, P:xi
_ (underscore) characters in Avaki names API:81
{ } curly brackets in command syntax AD:xv, C:xv, P:xi
| vertical bars in command syntax AD:xv, C:xv, P:xi

A

About My Domain screen AD:98
AbstractTransformer class P:243
AbstractTransformerFactory class P:244
access control in view models S:74
access control lists, See ACLs
accessibleDBOp SOAP operation API:42
accessibleDS SOAP operation API:36
accessiblePath SOAP operation API:19
accounts for grid users, See users AD:167
ACLs
 about O:45
 adding users and groups AD:243

ACLs (*continued*)

 defined AD:349, API:83, C:307, O:61, P:289, S:175
 deny permissions ineffective for owners, admins O:46
 displaying AD:237, C:186
 for database operations P:22, P:36
 for SQL views P:46
 granting or denying access to everyone O:44
 in grid groups O:43
 interpreting O:48
 modifying AD:239, C:41, S:97
 on cached objects O:50
 on new Avaki shares AD:261
 on new files O:49
 ownership AD:242, O:46
 permissions in AD:242, AD:307
 removing users and groups from AD:242
 sample O:45
 setting for a grid object AD:171
 setting for database operations P:14
 using interconnect IDs to add users and groups to AD:304
Active Directory AD:148
 domain users group AD:157, AD:159, C:155
 See also authentication services, LDAP AD:148
addInputParameter JavaScript method for data service
 plug-ins P:202
addInputStream JavaScript method for data service
 plug-ins P:202
ad-hoc queries
 as web services
 AdHocDBOPExecutionParams complex type API:13
 executeAdHocDBOp SOAP operation API:43
 executeAdHocDBOpWithOutput SOAP operation API:44
 executeAdHocDBOpWithOutputAttach SOAP
 operation API:46
 executeAdHocDBOpWithOutputString SOAP
 operation API:47

ad-hoc queries (*continued*)

- code samples API:74
- defined AD:349, API:83, C:307, O:61, P:289, S:175
- enabling C:66
- enabling on a database connector P:4
- executing C:63
- on virtual database
 - executing C:282
 - parameter types, specifying C:283
 - parameter types, specifying C:65
 - using JDBC driver to run API:69, API:74
- AdHocDBOPExecutionParams complex type API:13
- administrative user accounts, setting up AD:44
- Administrators group O:43
 - about AD:45
 - permissions for AD:240
- admission policies AD:332
 - about AD:85
 - adding C:91
 - creating AD:87
 - deleting AD:88, C:97
 - displaying C:114
 - displaying Windows domains for C:114
 - setting default policies C:112
 - setting Windows domain info C:111
 - unsetting Windows domains for C:116
- aggregate functions S:110
 - in SQL statements, aliasing column names for P:25
- Aggregate operator S:108
- AIX requirements AD:3
- algorithms for join operator S:155
- aliases
 - for column names P:25, S:42
 - aliases for GDC machines AD:10
- Allow permission in ACL AD:243, O:48
- angle brackets in command syntax AD:xvi, API:vii, C:xv, P:xi
- Apache Ant for compiling data service plug-ins P:184
- Apache Axis API:5
 - data catalog example API:18
 - data service example API:35
 - database operations example API:42
- APIs
 - data catalog API:18
 - data services API:34
 - database operations API:40
 - for data service plug-ins
 - about P:185
 - distributed transaction API P:188
 - general data service API P:186
 - TrAX (Transformation API for XML) P:243
 - web services
 - about API:2
 - data service API:35
 - reference API:11
- AROMValue parameters P:212
- As is permission in ACL AD:242
- ASE, see Sybase ASE

- asterisks in command syntax AD:xvi, C:xv, P:xi
- attribute --delete command C:19
- attribute --list command C:19
- attribute --update command C:21
- attributes
 - about AD:245
 - configuring for SQL views P:44
 - creating AD:248
 - and modifying S:100
 - defined AD:349, API:83, C:307, O:61, P:289, S:175
 - deleting AD:254, C:19
 - displaying AD:246
 - displaying details about C:19
 - ldap/importOnDemand AD:158
 - nis/importOnDemand AD:164
 - of cache services C:291
 - of grid servers C:290
 - of patches C:290
 - searching on AD:233
 - setting values AD:252, C:21
 - system attributes AD:248
 - types of AD:250, C:22, S:101
 - user-defined attributes AD:248
 - who can edit S:101
- audit logging
 - about O:14
 - configuring AD:319
 - events captured by AD:322
- audit logs AD:108
- authentication in Avaki O:41
 - using AvakiPrincipal API:13
- authentication services
 - configuring default groups C:218
 - configuring default users C:220
 - configuring GIDs C:217, C:221
 - configuring UIDs C:219, C:222
 - defined AD:349, API:83, C:307, O:61, P:289, S:175
 - deleting AD:166
 - displaying information about AD:166
 - grid, creating groups on C:141
- LDAP
 - adding schedule exclusions for refreshing C:152
 - adding search bases C:160
 - deleting authentication services C:153
 - deleting import schedules C:154
 - deleting search bases C:160
 - displaying information about C:157
 - enabling users C:250
 - importing groups from AD:159, C:155
 - importing users from AD:157, C:155
 - integrating into the grid AD:148, C:157
 - listing import schedules C:158
 - scheduling refreshes AD:185
 - scheduling user imports C:149
 - setting page size for imports AD:145
 - updating C:160

authentication services (*continued*)

- NIS
 - adding schedule exclusions for refreshing C:180
 - deleting authentication services C:181
 - deleting import schedules C:181
 - displaying info about C:183
 - enabling users C:250
 - importing groups from AD:165, C:182
 - importing users from AD:164, C:182
 - integrating into the grid AD:162, C:184
 - listing import schedules C:184
 - scheduling user imports C:177
 - updating C:185
- refreshing imported accounts AD:185
- specifying for JDBC connections API:69
- types O:41
 - specifying for JDBC connections API:69
- authentication using AvakiPrincipal API:9
- auto-restart
 - about AD:37, C:5
 - configuring for a DGAS C:5
 - configuring for a GDC AD:38, C:9
 - configuring for a grid server AD:51, C:9
 - configuring for a proxy server C:12
 - configuring for a share server C:15
- avaki attribute --delete C:19
- avaki attribute --list C:19
- avaki attribute --update C:21
- avaki backup C:23
- avaki cache --evict C:24, C:32
- avaki cache --evict --all C:25
- avaki cache --evict --deleted C:26
- avaki cache --get C:27
- avaki cache --invalidate C:27
- avaki cache --invalidate --all C:28
- avaki cache --invalidate-dataservice-results C:29
- avaki cache --invalidate-dbop-results C:30
- avaki cache --list C:31
- avaki cache --set C:33
- avaki cache --unset C:34
- avaki cat C:35
- avaki categories --add-to-category C:35
- avaki categories --create C:36
- avaki categories --delete C:37
- avaki categories --describe C:37
- avaki categories --list C:38
- avaki categories --remove-from-category C:38, C:40
- avaki categories --set-description C:39
- avaki cd C:41
- avaki chmod C:41
- avaki chown C:42
- avaki client C:45
- avaki client --connect command AD:94
- avaki cp C:46
- avaki database operation --list-schedules C:88
- avaki dataservice --add-schedule C:48
- avaki dataservice --create C:52
- avaki dataservice --delete C:52
- avaki dataservice --delete-schedule C:54
- avaki dataservice --depends C:54
- avaki dataservice --execute C:55
- avaki dataservice --generate-sql view C:56
- avaki dataservice --info C:57
- avaki dataservice --list-schedules C:58
- avaki dataservice --update C:58
- avaki dbconn --allow-dbop-creation C:59
- avaki dbconn --delete C:61
- avaki dbconn --disallow-dbop-creation C:62
- avaki dbconn --execute C:63
- avaki dbconn --info C:64
- avaki dbconn --jdbc C:66
- avaki dbconn --provision-tables C:71
- avaki dbconn --show-tables C:73
- avaki dbconn --test C:72
- avaki dbop --add-schedule C:73
- avaki dbop --delete C:78
- avaki dbop --delete-schedule C:78
- avaki dbop --depends C:79
- avaki dbop --execute C:80
- avaki dbop --generate-sql view C:82
- avaki dbop --info C:83
- avaki dbop --jdbc C:83
- avaki dbop --jdbc --create-virtual-dbop C:87
- avaki dgas --add-admission-policy C:91
- avaki dgas --add-group-mapping C:92
- avaki dgas --add-user-mapping C:94
- avaki dgas --cifs-share-info C:95
- avaki dgas --clear-cached-credentials C:95
- avaki dgas --create-cifs-share C:96
- avaki dgas --delete-admission-policy C:97
- avaki dgas --delete-cache C:98
- avaki dgas --delete-cifs-share C:99
- avaki dgas --delete-group-mapping C:99
- avaki dgas --delete-user-mapping C:100
- avaki dgas --disconnect-cifs-client C:101
- avaki dgas --get-cache-size C:101
- avaki dgas --get-cache-statistics C:102
- avaki dgas --get-free-disk-space C:102
- avaki dgas --get-properties C:103
- avaki dgas --get-property C:103
- avaki dgas --get-property-list C:104
- avaki dgas --initialize C:104
- avaki dgas --list-cifs-clients C:105
- avaki dgas --list-cifs-shares C:105
- avaki dgas --list-group-mappings C:106
- avaki dgas --list-user-mappings C:106
- avaki dgas --modify-cifs-share C:107
- avaki dgas --read-log-properties C:107
- avaki dgas --reset-cache-statistics C:108
- avaki dgas --save-cache C:109
- avaki dgas --self-map C:109
- avaki dgas --self-unmap C:111
- avaki dgas --set-admission-policy-domain C:111
- avaki dgas --set-default-admission-policy C:112
- avaki dgas --set-property C:113

- avaki dgas --show-admission-policies C:114
- avaki dgas --show-admission-policy-domain C:114
- avaki dgas --sync-cache C:115
- avaki dgas --unset-admission-policy-domain C:116
- avaki dgas --unset-property C:117
- Avaki directories, See directories, Avaki
- avaki directory --add-schedule C:117
- avaki directory --cache C:122
- avaki directory --delete-schedule C:122
- avaki directory --do-not-cache C:126
- avaki directory --list-schedules C:126
- avaki domain --create C:127
- avaki domain --disconnect C:127
- avaki domain --info C:128
- avaki domain --interconnect C:128
- Avaki domains, See domains, Avaki
- Avaki EII software
 - overview O:1
 - typical deployment O:17
- avaki executionservice --info C:129
- avaki executionservice --set C:129
- avaki file --add-schedule C:130
- avaki file --cache-on-demand C:134
- avaki file --delete-schedule C:135
- avaki file --do-not-cache C:136
- avaki file --list-schedules C:136
- avaki file --pin C:137
- Avaki functions S:73
- Avaki Functions menu S:74
- avaki group --add --user C:138
- avaki group --create C:141
- avaki group --delete C:143
- avaki group --delete --user C:144
- avaki group --info C:145
- avaki group --list-user C:147
- avaki help C:148
- avaki id C:149
- Avaki installation directory AD:350, API:84, C:308, O:62, P:290, S:176
- avaki ldap --add-schedule C:149
- avaki ldap --delete C:153
- avaki ldap --delete-schedule C:154
- avaki ldap --import C:155
- avaki ldap --info C:157
- avaki ldap --integrate C:157
- avaki ldap --list-schedules C:158
- avaki ldap --searchbase C:160
- avaki ldap --update C:160
- avaki ln C:161
- avaki locks --clear C:163
- avaki locks --list C:164
- avaki login C:164
- avaki logout C:165
- avaki ls C:166
- avaki mkdir C:167
- avaki monitor --add C:167
- avaki monitor --clear C:168
- avaki monitor --create C:170
- avaki monitor --delete C:171
- avaki monitor --list C:172
- avaki monitor --result C:172
- avaki monitor --start C:173
- avaki monitor --stop C:174
- avaki mv command C:176
- avaki nis --add-schedule C:177
- avaki nis --delete C:181
- avaki nis --delete-schedule C:181
- avaki nis --import C:182
- avaki nis --info C:183
- avaki nis --integrate C:184
- avaki nis --list-schedules C:184
- avaki nis --update C:185
- avaki passwd C:185
- avaki permissions C:186
- Avaki perspective in Studio S:13
- avaki plugin command P:184
- avaki plugin --generate C:187
- avaki proxy --add C:191
- avaki proxy --delete C:191
- avaki proxy --list C:192
- avaki pwd C:193
- avaki replica --add C:193
- avaki replica --config command C:193
- avaki replica --delete C:194
- avaki replica --info C:194
- avaki replicate --synch C:195
- avaki rm C:195
- Avaki rowset XML
 - class-name element P:279
 - column-display-size element P:279
 - column-index element P:279
 - core schema P:277
 - rowset-specific schema P:279
 - sample schema P:280
 - schema overview P:277
- avaki schedule --delete C:196
- avaki schedule --info C:197
- avaki schedule --list C:197
- avaki schedule --print-iterations C:198
- avaki scheduleexclusion --create --custom C:198
- avaki scheduleexclusion --create --daily C:199
- avaki scheduleexclusion --create --monthly C:201
- avaki scheduleexclusion --create --weekly C:203
- avaki scheduleexclusion --create --yearly C:205
- avaki scheduleexclusion --delete C:207
- avaki scheduleexclusion --info C:208
- avaki scheduleexclusion --list C:209
- avaki search (execute) C:211
- avaki search --create command C:209
- avaki search --delete C:210
- avaki search --get-rehash-level C:212
- avaki search --info C:214
- avaki search --rehash C:215
- avaki search --set-rehash-level C:215
- avaki security --config C:216
- avaki security --default-gid C:217

- avaki security --default-group C:218
- avaki security --default-uid C:219
- avaki security --default-user C:220
- avaki security --gid C:221
- avaki security --info C:222
- avaki security --uid C:222
- avaki server --dgas --connect C:223
- avaki server --dgas --destroy C:224
- avaki server --dgas --stop C:225
- avaki server --grid --connect C:225
- avaki server --grid --destroy C:226
- avaki server --grid --stop C:227
- avaki server --proxy C:228
- avaki server --share --connect C:228
- avaki server --share --disconnect C:229
- avaki server --share --stop C:230
- Avaki servers
 - distribution of data catalog among O:38
 - hardware and operating system requirements for O:16
 - qualified names for O:32
- avaki share --add-rehash-schedule C:231
- avaki share --add-share-servers C:232
- avaki share --create C:235
- avaki share --delete-rehash-schedule C:236
- avaki share --disconnect C:238
- avaki share --get-local-path C:238
- avaki share --get-status C:239
- avaki share --list-rehash-schedules C:239
- avaki share --list-share-servers C:240
- avaki share --rehash C:240
- avaki share --remove-share-servers C:241
- avaki share --set-local-path C:241
- avaki share --set-share-servers C:242
- avaki share --set-status C:243
- avaki share --update-share-servers C:244
- Avaki shares
 - about AD:257
 - adding schedule exclusions for rehashes C:234
 - adding share servers C:232
 - behavior during failover AD:112
 - bringing on line AD:286
 - changing configuration of AD:266
 - changing encryption levels AD:279
 - changing permissions AD:239
 - changing the owner AD:261
 - configuring exclusions for refresh schedules AD:274
 - copying into, out of, and within AD:213
 - creating AD:258, C:235
 - defined AD:350, API:84, C:308, O:62, P:290, S:176
 - deleting C:195
 - disconnecting C:238
 - disconnecting permanently AD:287
 - forcing refresh AD:262
 - icon for O:29
 - linking AD:217
 - local paths for, obtaining C:238
 - modifying load balance factor C:244
 - moving AD:210
- Avaki shares (*continued*)
 - moving source directories AD:283
 - naming of files and directories in AD:207
 - online status, setting C:243
 - organizing O:37
 - permissions on new files in O:49
 - refresh schedules
 - adding C:231
 - deleting C:236
 - listing C:239
 - refresh schedules for AD:266
 - refreshing C:240
 - removing entries from refresh schedules AD:278
 - removing share servers from AD:265
 - renaming AD:212
 - setting load balancing factor AD:280
 - setting local paths C:241
 - setting names AD:260
 - share servers
 - listing C:240
 - removing C:241
 - replacing C:242
 - shutting down AD:287, C:238
 - status, displaying C:239
 - taking off line AD:285
 - uploading files to AD:282
 - with multiple share servers AD:263
 - write access and user accounts AD:12
 - See also share servers
- avaki shell C:245
- avaki sql view --delete C:246
- avaki sql view --get-description C:246
- avaki sql view --set-description C:247
- avaki status C:248
- Avaki Studio
 - about O:9, O:17, S:1
 - Avaki perspective, about S:16
 - defined AD:350, API:84, C:308, O:62, P:290, S:176
 - getting started S:9
 - installing in Windows AD:24
 - limitations of data services created in P:78, S:3
 - log properties file for AD:317
 - metadata models, See metadata models
 - operators S:5, S:107
 - projects, creating S:13
 - requirements for running AD:3
 - setting system properties for AD:129
 - starting S:9
 - time required to upgrade AD:341
 - view models
 - about S:2
 - configuring input sources S:43
 - creating S:29
 - deploying as data services S:50
 - sample workflow for S:29
 - testing S:49
 - workflow S:25

avaki upgrade C:249
 avaki upgrade --info C:250
 avaki user C:250
 avaki user --create C:251
 avaki user --db-mapping --add C:252
 avaki user --db-mapping --delete C:253
 avaki user --db-mapping --list C:255
 avaki user --delete C:257
 avaki user --info C:258
 avaki user --list-group C:258
 avaki view --add-schedule C:259
 avaki view --create --database C:263
 avaki view --create --data-service C:266
 avaki view --create --file C:267
 avaki view --delete C:272
 avaki view --delete-schedule C:272
 avaki view --depends C:272
 avaki view --garbage-collect C:273
 avaki view --info C:274
 avaki view --list-schedules C:274
 avaki view --regenerate C:273
 avaki view --set-property C:275
 avaki view --update C:279
 avaki virtualdatabase --allow-dbop-creation C:280
 avaki virtualdatabase --disallow-dbop-creation C:281
 avaki virtualdatabase --execute C:282
 avaki virtualdatabase --show-tables C:283
 avaki virtualeschema --deploy C:285
 avaki virtualeschema --undeploy C:286
 avaki whoami C:286
 Avaki_JDBCStandAlone.jar file API:66
 Avaki_JDBCStandAlone_Minus3rd.jar file API:66
 AvakiAPI.disco file API:3
 AvakiAPIDocLit.wsdL file API:3
 AvakiAPIRpeEnc.wsdL file API:3
 AvakiAPIWithMIMEDocLit.wsdL file API:3
 AvakiAPIWithMIMERpeEnc.wsdL file API:3
 avakijdbc.properties file API:67
 AvakiPrincipal complex type API:13
 Axis, See Apache Axis

B

backup command C:23
 backups on Avaki servers AD:113
 batch mode, JDBC API:77
 configuring database operations for P:27, P:250
 bindings.xml file
 copying during upgrade AD:344
 on grid servers AD:50
 on proxy servers AD:300
 on share servers AD:59
 block size file attribute C:290
 blocks file attribute C:290
 bootstrapping
 in Unix AD:16
 in Windows AD:23
 brackets, See curly brackets, square brackets, angle brackets

browsers, See web browsers
 build.xml file for data service plug-ins P:184, P:197
 BusinessObjects software unable browse Avaki objects with
 underscores API:81

C

cache --evict --all command C:25
 cache --evict command C:24, C:32
 cache --evict --deleted command C:26
 cache --get command C:27
 cache --invalidate --all command C:28
 cache --invalidate command C:27
 cache --invalidate-dataservice-results command C:29
 cache --invalidate-dbop-results command C:30
 cache --list command C:31
 cache services
 about P:119
 associating with data grid access servers P:113
 associating with grid servers P:111
 coherence windows P:107
 configuring P:116
 configuring per file P:117
 defined AD:351, API:85, C:309, O:63, P:291, S:177
 disassociating from data grid access servers P:114
 disassociating from grid servers P:112
 evicting cached files and directories P:135
 invalidating cached items P:136
 listing P:116
 listing cached data services P:163
 listing cached database operations P:148
 listing cached virtual database operations P:148
 listing pinned files and directories P:134
 on-demand caching P:119
 on-demand caching of database operation and data service
 results P:108
 on-demand caching of files P:107
 overriding default settings P:117
 pinning data services P:152
 pinning database operations P:139
 pinning files and directories P:120
 pinning virtual database operations P:139
 scheduled caching P:119
 scheduled caching of database operation and data service
 results P:109
 scheduled caching of files P:107
 tagging files and directories P:129
 unmarking cached items P:135
 unscheduling cached files and directories P:135
 viewing details about P:116
 See also caches and caching
 cache --set command C:33
 cache --unset command C:34
 caches
 adding schedules for data services C:48
 adding schedules for database operations C:73
 adding schedules for directories C:117
 adding schedules for files C:130

caches (*continued*)

- bad port, properties for AD:141
- configuring associated server or user C:33
- configuring threads for AD:131
- data service plug-in, properties for AD:137
- deleting schedules for data services C:54
- deleting schedules for database operations C:78
- deleting schedules for directories C:122
- deleting schedules for files C:135
- DGAS
 - clearing user credentials AD:117
 - clearing user credentials from C:95
 - configuring block size for reads AD:81
 - configuring frags per block for reads AD:81
 - configuring location of AD:73
 - controlling cache size AD:124
 - deleting files and directories AD:119
 - deleting objects from C:98
 - displaying cache statistics C:102
 - displaying current size C:101
 - displaying free disk space on cache machine C:102
 - forcing a refresh AD:121
 - managing AD:117
 - mapping cache AD:336
 - resetting statistics C:108
 - saving a copy C:109
 - saving copies AD:120
 - setting remote caches for AD:90
 - syncing AD:121
 - viewing and resetting statistics AD:123
 - warming and updating C:115
- displaying associated server or user C:27
- displaying tagging information C:31
- for tables in virtual database, property for AD:144
- listing schedules for data services C:58
- listing schedules for database operations C:88
- listing schedules for directories C:126
- listing schedules for files C:136
- local S:182
- local and remote API:63
- marking directories for no caching C:126
- marking files for no caching C:136
- remote object stub, properties for AD:144
- schedule exclusion, properties for AD:144
- scheduled caching S:184
- settable attributes of C:291
- setting invalidate queue for AD:135
- setting local directory for AD:135
- setting remote caches for command clients AD:95
- uncoupling associated server or user C:34

See also cache services and caching

caching

- about O:13
- and JDBC programs O:55
- benefits to performance O:54
- configuring ACLs for O:50

caching (*continued*)

- configuring Avaki clients for O:55
- data service results P:108, S:51
 - tagging for on-demand caching P:159
- database operations P:108
- defined AD:351, API:85, C:309, O:63, P:291, S:177
- files O:56, P:107
- JDBC and caching of database operation results API:62
- local AD:356, API:90, C:314, O:14, O:53, O:68
- local vs. remote O:59
- local, defined P:296
- of data service results O:57
- of database operation results O:57
- on DGAS O:54
- on-demand AD:357, API:91, C:315, O:69, P:296, S:183
- permissions and access control O:59
- remote O:14, O:53
 - defined AD:357, API:91, C:315, O:69, P:297, S:184
- scheduled AD:357, API:91, C:315, O:69
- scheduled, defined P:297
- turning off for specified files and directories P:132
- See also caches and cache services
- callable statements API:72, API:73
- case sensitivity in Avaki naming AD:206
- cat command C:35
- catalog browser S:18
- categories
 - about AD:221
 - adding objects to AD:226, C:35, S:105
 - adding SQL views P:47
 - and permissions AD:222–AD:223
 - browsing AD:222
 - contents of S:18
 - creating AD:224, C:36
 - default, contents of S:18
 - defined AD:351, API:85, C:309, O:63, P:291, S:177
 - deleting AD:230, C:37, S:106
 - displaying S:104
 - for logging AD:318
 - icon for O:29
 - listing categories in domain C:38
 - managing S:103
 - permissions in O:48
 - removing objects from AD:228, C:38, C:40, S:106
 - setting descriptions for C:39
 - showing descriptions C:37
 - using to organize data O:36
 - using to solve access problems O:49
- categories --add-to-category command C:35
- categories --create command C:36
- categories --delete command C:37
- categories --describe command C:37
- Categories directory O:35
- categories --list command C:38
- categories --remove-from-category command C:38, C:40
- categories --set-description command C:39
- cd command C:41
- certificates, SSL, See SSL certificates

- change time file attribute C:290
- characters
 - in column aliases in database operations S:42
 - in command syntax AD:xv, C:xv
 - in cron schedules C:298
 - in domain names, restrictions on AD:41
 - in JavaScript identifiers S:42
 - in metadata model names, restrictions on S:91
 - in names of Avaki objects, restrictions on AD:207
 - wildcards in searches AD:235
- CHARSET JDBC property for ASE and IQ AD:7
- chmod command C:41
- chmod SOAP operation API:19
- chown command C:42
- chown SOAP operation API:20
- chunk size for sorting, controlling AD:139, S:76
- CIFS
 - accessing data grid through AD:338
 - releasing CIFS ports on a DGAS AD:66
- CIFS clients
 - defined AD:351, API:85, C:309, O:63, P:291, S:177
 - disconnecting C:101
 - displaying connected clients C:105
 - requirements for O:17
 - setting up AD:93
- CIFS shares
 - accessing AD:203
 - creating AD:125, C:96
 - defined AD:351, API:85, C:309, O:63, P:291, S:177
 - deleting C:99
 - displaying connected clients C:105
 - displaying information about C:95
 - listing C:105
 - managing AD:125
 - mapping to a network drive AD:204
 - modifying C:107
- class element P:261
- class-name element P:279
- classpath, configuring for JDBC drivers API:67
- client attribute caching AD:336
- client command C:45
- client system properties AD:128
- clients
 - about O:17
 - hardware and operating system requirements for O:16
 - message timeout properties for AD:133
 - setting size of write invalidation queue of cache for AD:136
 - setting system properties for AD:129
 - See also Avaki Studio, CIFS clients, command clients, NFS clients, web clients, WS clients
- code samples
 - ad-hoc queries API:74
 - data catalog API API:18
 - data services API API:35
 - database operations API API:42
 - Java data service plug-ins P:190
 - JDBC batch mode API:77
 - using JDBC drivers API:77
- coherence window cache attribute C:291
- coherence window property, remote AD:141
- coherence windows for caching P:107
- coherenceWindow element P:261
- colors in Studio display, setting S:23
- column-display-size element P:279
- column-index element P:279
- columns
 - aliasing P:25
 - combining with Projection operator S:46
 - from input elements, menus of S:71
 - from input result sets, accessing S:68
 - name property S:60
 - precision property S:61
 - scale property S:61
 - type property S:61
- com.avaki.badPortCacheSize system property AD:141
- com.avaki.badPortExpiration system property AD:141
- com.avaki.cache.cacheDir system property AD:135
- com.avaki.cache.maxReaderThreads system property AD:131
- com.avaki.cache.writeInvalidationQueueSize system property AD:136
- com.avaki.content.encryptionLevel system property AD:139
- com.avaki.dataservice.pluginCacheSize system property AD:137
- com.avaki.dataservice.styleSheetCachePoolSize system property AD:137
- com.avaki.dataservice.styleSheetCacheSize system property AD:137
- com.avaki.DBOProtocolSoTimeout system property AD:134
- com.avaki.generatedXMLIndentSize AD:142
- com.avaki.HttpPort system property AD:140
- com.avaki.HttpsPort system property AD:140
- com.avaki.jobStatusExpiration system property AD:145
- com.avaki.lasInvoker.cacheSize system property AD:144
- com.avaki.lasInvoker.poolSize system property AD:144
- com.avaki.ldap.resultPageSize system property AD:145
- com.avaki.maxActiveCachables system property AD:136
- com.avaki.mux.channelSoTimeout system property AD:135
- com.avaki.mux.connectTimeout system property AD:134
- com.avaki.mux.maxParallelChannels system property AD:142
- com.avaki.mux.maxWriteChunk system property AD:142
- com.avaki.mux.sendBufferSize system property AD:143
- com.avaki.proxy.retryDelay system property AD:133
- com.avaki.proxy.retryTimeout system property AD:133
- com.avaki.proxyIOProtocolSoTimeout system property AD:134
- com.avaki.proxyKeepAliveParams system property AD:140
- com.avaki.queryEngine.sortChunkSize AD:139
- com.avaki.remoteconfig.coherenceWindow system property AD:141
- com.avaki.result.gcInterval system property AD:136
- com.avaki.retryDelay system property AD:133
- com.avaki.retryTimeout system property AD:133
- com.avaki.rmiRegistrySoTimeout system property AD:134
- com.avaki.rpcTimeout system property AD:134
- com.avaki.scheduleExclusionCacheExpiration system property AD:145

com.avaki.scheduleExclusionCacheSize system property AD:145

com.avaki.shareIOProtocolSoTimeout system property AD:134

com.avaki.shareReadBufferSize system property AD:138

com.avaki.shareReadbufPoolSize system property AD:138

com.avaki.shareServerCircularLinkChecking system property AD:138

com.avaki.shareServerThreadPoolSize system property AD:138

com.avaki.vaultStateCacheSize system property AD:137

com.avaki.VirtualDbTableCacheSize system property AD:144

com.sybase.avaki.tdsPort system property AD:50, AD:145, API:71

command clients

- connecting C:45
- defined AD:352, API:86, C:310, O:64, P:292, S:178
- disconnecting C:45
- installing in Windows AD:24
- installing on Unix AD:18
- obtaining information about C:45
- setting up AD:94

commands

- listing C:148
- syntax conventions for AD:xv, C:xiv, P:x
- viewing online usage information C:148

compatibility properties, setting for Windows 2003 AD:22

complex types API:12

- AdHocDBOPExecutionParams API:13
- AvakiPrincipal API:13
- DataCatalogAttribute API:14
- DataCatalogEntry API:15
- DataCatalogPermission API:15
- DataServiceExecutionParams API:16
- DBOPExecutionParams API:16
- SearchQuery API:17
- SearchResult API:17

condition field for Iterator operators S:151

connect ports

- default AD:6, AD:9, AD:10
- defined AD:352, API:86, C:310, O:64, P:292, S:178
- for DGAS C:224
 - changing AD:74
- for GDCs C:127
 - changing AD:50
- for grid servers C:226
 - changing AD:50
- for proxy servers C:228
 - changing AD:300
- for share servers C:229
 - changing AD:60

connectinfo.txt file AD:131

connection pooling S:36

connection properties

- for JDBC drivers API:68
- for XA drivers C:70, P:7, S:37

connection strings

- for databases AD:3
- for JDBC drivers API:71

- connections in view models, creating S:57
- console view S:22, S:50
- conventions
 - for command syntax C:xiv
 - for commands AD:xv
 - for screen examples AD:xv, API:vi, C:xv, P:xi
- cp command C:46
- cron expressions in schedules AD:185, AD:267, AD:273
- cron schedules
 - configuring C:297
 - values for C:298
- cross-domain messaging
 - disabling AD:313
 - enabling AD:311
- curly brackets in command syntax AD:xv, C:xv, P:xi
- CurrentUser functions S:74
- Custom operator S:111
 - example S:114
- custom types API:12

D

- data access O:11
 - using WS API API:2
- data catalog
 - about O:27
 - defined AD:352, API:86, C:310, O:64, P:292, S:178
 - distribution among Avaki servers O:38
 - names of objects in O:24
 - organizing O:33
 - Avaki shares O:37
 - using categories O:36
 - using links O:36
 - ownership of objects in O:46
 - top-level directories O:32
 - types of entries O:6
- data catalog API API:18
- data catalog SOAP operations API:18
 - accessiblePath API:19
 - chmod API:19
 - chown API:20
 - fileRead API:21
 - fileReadAttach API:21
 - fileReadString API:22
 - fileWrite API:23
 - getAttributes API:23
 - getSystemAttributes API:24
 - getUserAttributes API:24
 - listDomains API:25
 - listSearches API:25
 - ls API:26
 - lsSize API:26
 - mkdir API:27
 - mkdirParents API:27
 - mkdirParentsServer API:28
 - mkdirServer API:29
 - mv API:29
 - permissions API:30

- data catalog SOAP operations (*continued*)
 - removeAttribute API:31
 - rm API:31
 - search API:32
 - setAttribute API:32
 - tester API:33
 - whoami API:33
- data catalog view S:18
- data expiration intervals P:108, S:51
- data grid access servers
 - associating with cache services P:113
 - disabling auto-restart C:8
 - disassociating from cache services P:114
 - enabling auto-restart C:5
 - registering C:5
 - starting C:5, C:6
 - stopping C:7, C:8
 - unregistering C:8
 - See also DGAS
- data grids
 - about O:1
 - defined AD:355, API:89, C:313, O:67, P:294, S:181
 - typical deployment O:17
- data integration O:21, O:23
- data integrity and HTTPS API:8
- data representation O:11
- data security O:10
- data service plug-ins
 - about P:76, P:175
 - addInputParameter JavaScript method P:202
 - addInputStream JavaScript method P:202
 - build.xml file P:184, P:197
 - choice of Java, JavaScript, or XSLT P:176
 - closing streams P:186
 - command for generating C:187
 - configuring P:81
 - creating in Java with the Plug-in Wizard P:183
 - creating in JavaScript P:200
 - creating in XSLT P:180
 - DbopGroupWorkUnit class P:189
 - DbopPipeWorkUnit class P:190
 - defined AD:352, API:86, C:310, O:64, P:292, S:179
 - examples
 - DBOP and CSV merge Java plug-in P:193
 - distributed transaction Java plug-in P:191
 - rowset input and output Java plug-in P:192
 - Execute JavaScript function P:203
 - input sources and output streams P:177
 - InputSource interface P:186
 - JAR files for P:180
 - logging errors P:196
 - manifest files for P:180, P:197
 - modularity and reusability of P:175
 - parameters
 - about P:178
 - specifying for Java plug-ins C:188
 - specifying for XSLT plug-ins P:181
- data service plug-ins (*continued*)
 - ParameterSpec interface P:187
 - Plugin interface P:186
 - prerequisites for writing in Java P:183
 - relationship to .js files in Studio S:11
 - RowSetFactory class P:188
 - setOutputStream JavaScript method P:202
 - StreamingRowSet interface P:187
 - using Java classes and interfaces in JavaScript plug-ins P:200
 - when to use P:78, S:3
 - XAWorkHandler class P:189
 - XAWorkUnit interface P:189
- data service XML schema
 - class element P:261
 - coherenceWindow element P:261
 - dataService element P:262
 - description element P:263
 - initParameter element P:263
 - inputParameter element P:264
 - inputSource element P:265
 - inputStream element P:266
 - isList element P:266
 - jarurl element P:267
 - logicBox element P:268
 - name element P:269
 - outputStream element P:269
 - ref element P:270
 - target element P:270
 - type element P:270
 - urlLogicBox element P:271
 - value element P:272
 - values element P:272
- data services
 - about O:8, O:23, P:49, P:74
 - adding schedule exclusions C:51
 - and distributed transactions P:78
 - caching of results O:57, P:77
 - permissions O:50
 - caching results S:51
 - calling via JDBC API:72
 - components of P:76
 - configuring caching P:108
 - created in Avaki Studio, limitations of P:78, S:3
 - creating C:52, P:80, P:207
 - defined AD:352, API:86, C:310, O:64, P:292, S:178
 - deleted, purging from cache C:26
 - deleting schedules C:54
 - dependencies for S:22
 - deploying from view models in Avaki Studio S:50
 - displaying dependency lists C:54
 - displaying information about C:57, S:20
 - displaying status of C:248
 - evicting from cache P:164
 - execution services, configuring AD:109
 - generating schema for C:55, P:98
 - generating SQL views from C:56, P:100
 - importing descriptors P:92
 - input parameters, configuring P:84

- data services (*continued*)
 - input streams, configuring P:87
 - invalidating all in cache C:28
 - invalidating one in cache C:27
 - invalidating results in cache C:29
 - listing P:93
 - listing caching schedules for C:58
 - listing in cache P:163
 - location in categories S:18
 - marking for scheduled caching P:152
 - modifying C:58
 - modifying permissions AD:239
 - modifying settings P:94
 - names in data catalog O:24
 - nesting operations S:149
 - output streams, configuring P:86
 - provisioning web services as P:205
 - purging all from cache and unspooling C:25
 - purging one from cache and unspinning C:24
 - qualified names for O:31
 - refreshing cached results C:32
 - removing P:103
 - rowsets as input of P:275
 - rowsets as output of P:274
 - running C:55
 - sample workflow for S:29
 - scheduling for caching C:48
 - schema P:257
 - searching for AD:233
 - setting cache sizes for plug-ins AD:137
 - setting up to run distributed transactions P:80
 - specifying grid servers P:213
 - specifying input parameters P:207
 - specifying input streams P:208
 - specifying output streams P:208
 - specifying plug-ins P:207
 - tagging for on-demand caching P:159
 - testing P:102, P:214
 - unscheduling P:164
 - using for distributed transactions O:25
 - viewing P:98
 - viewing dependencies P:97
 - writing your own vs. using Avaki Studio O:24
 - See also data service plug-ins
 - See also view models
- data services API API:34
- data services SOAP operations API:34
 - accessibleDS API:36
 - executeDS API:36
 - getDSOutput API:37
 - getDSOutputAttach API:38
 - getDSOutputString API:38
 - getDSParameters API:39
 - isDSAvakiXML API:40
 - listDSs API:40
- data structures, SOAP complex types API:12
- data type mappings for SQL views P:39
- data types
 - for JDBC API:76
 - mapping
 - about type mapping files C:301
 - command to specify mapping file C:68
 - format of type mapping files C:301
 - inconsistencies C:302
 - logging of mapping decisions C:304
 - setting source data type C:302
 - specifying for ad-hoc query parameters C:65
 - specifying for database operation parameters C:85
 - specifying for parameters for ad-hoc queries on the virtual database C:283
 - specifying for virtual database operation parameters C:88
 - See also type
- database connectors
 - about O:22, P:3
 - adding groups P:16
 - adding users P:15
 - configuring advanced settings P:247
 - configuring JDBC driver JAR file path P:247
 - configuring permissions C:59, C:62
 - creating C:66, P:3, S:31
 - data type mappings for P:39
 - defined AD:353, API:87, C:311, O:65, P:293, S:179
 - deleting C:61
 - displaying information about C:64
 - displaying SQL views provisioned from C:73
 - editing S:38
 - executing ad-hoc queries C:63, C:66
 - finding in catalog S:38
 - getting information about through JDBC API:75
 - JDBC fetch size P:5
 - location in categories S:18
 - managing SQL views P:20
 - modifying C:66, P:8
 - provisioning SQL views from C:71
 - removing P:21
 - removing groups P:18
 - removing users P:18
 - searching for AD:233
 - setting JDBC fetch size S:36
 - testing C:72, P:19
 - viewing P:8
 - viewing associated database operations P:13
- database drivers
 - copying during upgrade AD:341
 - tested with Avaki AD:3
- database identity mappings P:6, S:36
 - about AD:176
 - adding C:252
 - deleting AD:183, C:253
 - displaying AD:180
 - listing C:255
 - modifying AD:182
 - setting up AD:177
- database operation --list-schedules command C:88

database operation SOAP operations API:40

database operations

about O:7, O:22, P:1

access permissions P:22

adding schedule exclusions C:76

allowing groups to create P:16

allowing users to create P:15

caching of results API:62, O:57

permissions O:50

calling with JDBC API:73

calling with ODBC, JDBC, or SOAP P:38

configuring advanced settings P:247

configuring batch mode settings P:250

configuring caching P:108

configuring permissions C:59, C:62

configuring SQL calls P:251

configuring timeouts P:253

creating C:83, P:22, S:38

defined AD:353, API:87, C:311, O:65, P:293, S:179

deleted, purging from cache C:26

deleting C:78

deleting schedules C:78

dependencies for S:22

displaying dependency lists C:79

displaying information about C:83, S:20

displaying status of C:248

evicting from cache P:150

executing P:36

exposing results as SQL view P:34

generating schema for C:80, P:31

generating SQL views from C:82

invalidating all in cache C:28

invalidating one in cache C:27

invalidating results in cache C:30

listing caching schedules for C:88

listing in cache P:148

location in categories S:18

managing P:21

managing metadata P:30

marking for scheduled caching P:139

modifying C:83, P:28

modifying permissions AD:239

names in data catalog O:24

parameter types, specifying C:85, C:88

preventing groups from creating P:18

preventing users from creating P:18

purging all from cache and unscheduling C:25

purging one from cache and unpinning C:24

qualified names for O:31

refreshing cached results C:32

removing P:38

removing SQL views generated from P:35

restricting row output P:248

rowsets as output of P:274

running C:80

sample XML schema P:280

scheduling for caching C:73

searching for AD:233

database operations (*continued*)

setting JDBC fetch size for P:254

setting permissions P:14

SQL statements in C:86

tagging for on-demand caching P:146

transactional behavior of P:79

unscheduling P:150

uses of P:2

viewing P:13, P:28

viewing dependencies P:32

viewing details about P:29

See also virtual database operations

database operations API API:40

database service SOAP operations

accessibleDBOp API:42

executeAdHocDBOp API:43

executeAdHocDBOpWithOutput API:44

executeAdHocDBOpWithOutputAttach API:46

executeAdHocDBOpWithOutputString API:47

executeDBOp API:48

executeDBOpBytesInput API:49

executeDBOpGridFileInput API:50

executeDBOpWithOutput API:50

executeDBOpWithOutputAttach API:52

executeDBOpWithOutputString API:53

getDBOpOutput API:54

getDBOpOutputAttach API:55

getDBOpParameters API:56

getDBOpSchema API:56

getDBOpSchemaAttach API:57

getDBOpSchemaString API:58

getOutputString API:55

getSQL API:58

listDBConns API:59

listDBOps API:59

listDBOpsByDBConn API:60

database, virtual, See virtual database

DatabaseAdministrators group O:44

DatabaseMetaData interface API:75

databases

Avaki tools for working with O:21

connecting to P:3

for Avaki servers, backing up C:23

protecting O:9

schemas, viewing P:9

supported for connecting to Avaki AD:3, AD:5

DataCatalogAttribute complex type API:14

DataCatalogEntry complex type API:15

DataCatalogPermission complex type API:15

DataProviders group O:44

dataservice --add-schedule command C:48

dataservice --create command C:52

dataservice --delete command C:52

dataservice --delete-schedule command C:54

dataservice --depends command C:54

dataService element P:262

dataservice --execute command C:55

dataservice --generate-sql view command C:56

- dataservice --info command C:57
- dataservice --list-schedules command C:58
- dataservice --update command C:58
- DataServiceExecutionParams complex type API:16
- DB2, versions and JDBC drivers for use with Avaki AD:6
- dbconn --allow-dbop-creation command C:59
- dbconn --delete command C:61
- dbconn --disallow-dbop-creation command C:62
- dbconn --execute command C:63
- dbconn --info command C:64
- dbconn --jdbc command C:66
- dbconn --provision-tables command C:71
- dbconn --show-tables command C:73
- dbconn --test command C:72
- DBOPs, See database operations
- dbop --add-schedule command C:73
- dbop --delete command C:78
- dbop --delete-schedule command C:78
- dbop --depends command C:79
- dbop --execute command C:80
- dbop --generate-sql view command C:82
- dbop --info command C:83
- dbop --jdbc command C:83
- dbop --jdbc --create-virtual-dbop command C:87
- DBOPExecutionParams complex type API:16
- DbopGroupWorkUnit class for data services P:189
- DbopPipeWorkUnit class for data services P:190
- db-path option (DGAS) AD:74
- debug mode, enabling in an Avaki shell C:245
- delimiter character for JDBC schema names API:69
- Deny permission in ACL AD:243, O:47
- dependencies S:22
 - defined AD:353, API:87, C:311, O:65, P:293, S:179
 - listing for data services C:54, P:97
 - listing for database operations C:79, P:32
 - listing for view generators C:272, P:228
 - listing for virtual database operations P:59
- description element P:263
- descriptors for data services P:92
- development tools for web services
 - Apache Axis API:5
 - Microsoft Visual Studio API:5
 - SOAP::Lite API:5
 - VB .NET API:5
- DGAS
 - about AD:62
 - adding user self mappings C:109
 - admission policies AD:332
 - about AD:85
 - adding AD:87, C:91
 - deleting AD:88, C:97
 - displaying C:114
 - displaying Windows domains for C:114
 - admission policies
 - setting defaults C:112
 - setting Windows domains for C:111
 - unsetting Windows domains for C:116
- DGAS (*continued*)
 - changing permissions and ownership AD:334
 - CIFS access to data grid AD:338
 - clearing cached credentials AD:117, C:95
 - configuring associated cache service C:33
 - configuring location of internal caches AD:73
 - configuring to use nondefault ports AD:74, AD:75
 - configuring users and groups AD:67
 - connect port C:224
 - connecting to a domain AD:79, C:223
 - controlling cache size AD:124
 - create CIFS shares C:96
 - default name for AD:73
 - default users, groups, UIDs and GIDs AD:333
 - defined AD:352, API:86, C:310, O:64, P:292, S:178
 - deleting cached objects C:98
 - deleting CIFS shares C:99
 - deleting files and directories from cache AD:119
 - deleting user mappings C:111
 - destroying C:224
 - disconnecting CIFS clients C:101
 - displaying associated cache C:27
 - displaying cache size C:101
 - displaying cache statistics C:102
 - displaying connected CIFS clients C:105
 - displaying free disk space on cache machine C:102
 - displaying information about CIFS shares C:95
 - displaying property descriptions C:104
 - displaying property values C:103
 - dynamic and nondynamic properties AD:83
 - file locking in Unix, interference with AD:64
 - forcing cache to refresh AD:121
 - installing in Unix AD:18
 - installing in Windows AD:24
 - listing CIFS shares C:105
 - listing properties and their values C:103
 - managing cache AD:117
 - mappings
 - default, adding and displaying AD:71
 - for groups, adding AD:70, C:92
 - for groups, deleting C:99
 - for groups, displaying C:106
 - for users, adding AD:70, C:94
 - for users, deleting C:100
 - for users, displaying C:106
 - users and groups, per-DGAS AD:88
 - users, groups, and defaults, domain-wide AD:68
 - modifying CIFS shares C:107
 - NFS clients, not running with AD:64
 - NFS daemons, shutting down before running DGAS AD:66
 - per-DGAS user mappings AD:333
 - ports used by AD:9
 - preparing to start AD:65
 - properties file for AD:76, C:293
 - reading log properties C:107
 - releasing CIFS ports before running AD:66
 - resetting cache statistics C:108

DGAS (continued)

restarting AD:84
 saving a copy of the cache C:109
 saving copies of cache AD:120
 server logs AD:317
 setting a cache service AD:90
 setting location of state database AD:74
 setting properties AD:82, C:113
 setting up NFS clients AD:91
 starting AD:73, C:104
 stopping C:225
 syncing cache AD:121
 time required to upgrade AD:341
 uncoupling associated cache C:34
 unsetting properties C:117
 viewing and resetting cache statistics AD:123
 warming and updating the cache C:115
 See also data grid access servers

dgas --add-admission-policy command C:91
 dgas --add-group-mapping command C:92
 dgas --add-user-mapping command C:94
 dgas --cifs-share-info command C:95
 dgas --clear-cached-credentials command C:95
 dgas command
 example AD:74
 syntax AD:73

dgas --create-cifs-share command C:96
 dgas --delete-admission-policy command C:97
 dgas --delete-cache command C:98
 dgas --delete-cifs-share command C:99
 dgas --delete-group-mapping command C:99
 dgas --delete-user-mapping command C:100
 dgas --disconnect-cifs-client command C:101
 dgas --get-cache-size command C:101
 dgas --get-cache-statistics command C:102
 dgas --get-free-disk-space command C:102
 dgas --get-properties command C:103
 dgas --get-property command C:103
 dgas --get-property-list command C:104
 dgas --initialize command C:104
 dgas --list-cifs-clients command C:105
 dgas --list-cifs-shares command C:105
 dgas --list-group-mappings command C:106
 dgas --list-user-mappings command C:106
 dgas --modify-cifs-share command C:107
 dgas --read-log-properties command C:107
 dgas --register command C:5
 dgas --reset-cache-statistics command C:108
 dgas --save-cache command C:109
 dgas --self-map C:109
 dgas --self-unmap C:111
 dgas --set-admission-policy-domain command C:111
 dgas --set-default-admission-policy command C:112
 dgas --set-property command C:113
 dgas --show-admission-policies command C:114
 dgas --show-admission-policy-domain command C:114
 dgas --start command C:6

dgas --stop command C:7
 dgas --sync-cache command C:115
 dgas --unregister command C:8
 dgas --unset-admission-policy-domain command C:116
 dgas --unset-property command C:117

directories
 adding schedule exclusions for caching C:121, C:125
 adding to cache service C:122
 Avaki directories, defined AD:349, API:83, C:307, O:61,
 P:289, S:175
 Avaki installation API:84, P:290, S:176
 Avaki installation directory AD:350, C:308, O:62
 changing C:41
 changing ownership C:42
 changing permissions for AD:239
 copying AD:213, C:46
 creating AD:208, C:167
 deleted, purging from cache C:26
 deleting AD:219, C:195
 deleting caching schedules C:122
 displaying name of current directory C:193
 evicting from cache P:135
 exporting from the data grid AD:125
 home, creating AD:169
 icon for O:29
 invalidating all in cache C:28
 invalidating from cache P:136
 invalidating in cache C:27
 linking AD:217, C:161
 listing C:166
 listing schedules C:126
 listing those pinned for caching P:134
 marking for no caching C:126, P:132
 marking for scheduled caching P:120
 moving AD:210, C:176
 NFS-mounting AD:92
 permissions in O:48
 purging all from cache and unpinning C:25
 purging from cache and unpinning C:24
 refreshing in cache C:32
 renaming AD:212
 scheduling for caching C:117
 searching for AD:233
 setting ACLs for AD:171
 shared, See Avaki shares
 tagging for on-demand caching P:129
 temp, setting for grid servers AD:135
 top-level, described O:32
 unscheduling from cache P:135

directory --add-schedule command C:117
 directory --cache command C:122
 directory --delete-schedule command C:122
 directory --do-not-cache command C:126
 directory --list-schedules command C:126

disk space
 available, displaying for DGAS cache C:102
 requirements for Avaki software AD:4, AD:16

- distributed transactions
 - about O:25, P:78
 - API for executing P:188
 - configuring database connectors for P:7, S:36
 - defined AD:353, API:87, C:311, O:65, P:293, S:180
 - requirements for P:79
 - setting up P:80
 - supported DBMSes P:79
 - two-phase commit protocol P:79
 - DNS aliases for GDC machines AD:10
 - DNS name, setting for a server AD:32
 - document/literal web services API:3, API:5
 - documentation
 - Avaki, list of AD:xii, API:vi, C:xii, O:vi, P:viii, S:viii
 - for Eclipse Workbench S:12
 - domain --create command C:127
 - domain --disconnect command C:127
 - domain --info command C:128
 - domain --interconnect command C:128
 - domain names, defined AD:354, API:88, C:312, O:66, P:294, S:180
 - Domain Users group in Active Directory C:155
 - domains, Avaki
 - creating C:127
 - defined AD:350, API:84, C:308, O:62, P:290, S:176
 - disconnecting C:127
 - displaying information about AD:98
 - getting information about through JDBC API:75
 - interconnecting AD:289, C:128
 - joining together AD:289
 - naming AD:41, AD:354, API:88, C:312, O:66, P:294, S:180
 - obtaining information about C:128
 - planning before install AD:1
 - providers and consumers AD:289
 - remote, logging in to AD:201
 - specifying for JDBC connections API:69
 - DomainUsers group O:44
 - downstream variables menu S:71
 - downstream, defined S:3
 - drivers, See JDBC drivers
 - drivers directory AD:341
 - dynamic and nondynamic properties (DGAS) AD:83
 - dynamic user mappings
 - creating C:109
 - deleting C:111
- E**
- Eclipse Workbench S:12
 - EII, See enterprise information integration
 - elements
 - connecting S:57
 - descriptions of S:59
 - Input Source S:125
 - moving S:56
 - elements (*continued*)
 - names of S:58
 - operators S:5
 - properties dialogs S:58
 - Result S:164
 - selecting S:56
 - with red borders S:60
 - encryption and HTTPS API:8
 - encryption levels for Avaki shares
 - changing AD:279
 - displaying C:222
 - setting at share creation AD:261
 - encryption of grid objects AD:139
 - enterprise information integration, defined AD:354, API:88, C:312, O:66, P:294, S:180
 - equal signs in command syntax AD:xvi, C:xvi, P:xii
 - error handling S:143
 - errors in view models S:60
 - everyone group O:44, O:48
 - examples
 - conventions used in C:xv
 - data catalog web service API:18
 - data services API API:35
 - database operations API API:42
 - web services clients
 - Java API:9
 - Perl API:9
 - VB .NET API:9
 - exclusions, See schedule exclusions
 - execute inputs in parallel field for Iterator operators S:151
 - Execute JavaScript function for data service plug-ins P:203
 - executeAdHocDBOp SOAP operation API:43
 - executeAdHocDBOpWithOutput SOAP operation API:44
 - executeAdHocDBOpWithOutputAttach SOAP operation API:46
 - executeAdHocDBOpWithOutputString SOAP operation API:47
 - executeDBOp SOAP operation API:48
 - executeDBOpBytesInput SOAP operation API:49
 - executeDBOpGridFileInput SOAP operation API:50
 - executeDBOpWithOutput SOAP operation API:50
 - executeDBOpWithOutputAttach SOAP operation API:52
 - executeDBOpWithOutputString SOAP operation API:53
 - executeDS SOAP operation API:36
 - execution services
 - about AD:109, P:77, S:51
 - configuring AD:109, C:129
 - configuring threads for AD:131
 - defined AD:354, API:88, C:312, O:66, P:294, S:180
 - displaying information about C:129
 - executionservice --info command C:129
 - executionservice --set command C:129
 - executionServiceHint JDBC property API:70
 - exiting from an Avaki session C:165
 - expressions in operators S:4
 - expressions menu, using S:71
 - externalCacheService JDBC property API:62, API:70

F

- failover
 - defined AD:354, API:88, C:312, O:66, P:294, S:180
 - managing AD:112
 - setting up a secondary GDC AD:43
- fake_metadata JDBC connection property for ASE API:70
- FAKE_METADATA JDBC property for ASE AD:7
- federated data access AD:354, API:88, O:66, P:294, S:181
- fetch size, See JDBC fetch size
- file --add-schedule command C:130
- file --cache-on-demand command C:134
- file --delete-schedule command C:135
- file --do-not-cache command C:136
- file --list-schedules command C:136
- file locking AD:64
 - suppressing in NFS mount command AD:93
- file --pin command C:137
- file size attribute C:290
- fileRead SOAP operation API:21
- fileReadAttach SOAP operation API:21
- fileReadString SOAP operation API:22
- files
 - .amm files S:11
 - .avm files S:11
 - .js JavaScript files S:11
 - .jsi JavaScript include files S:11, S:75
 - sample S:115
 - adding schedule exclusions for caching C:133, C:140
 - Avaki_JDBCStandAlone.jar API:66
 - Avaki_JDBCStandAlone_Minus3rd.jar API:66
 - avaki_studio.properties AD:129
 - AvakiAPI.disco WSDL discovery file API:3
 - AvakiAPIDocLit.wsdl API:3
 - AvakiAPIRpcEnc.wsdl API:3
 - AvakiAPIWithMIMEDocLit.wsdl API:3
 - AvakiAPIWithMIMERpcEnc.wsdl API:3
 - avakijdbc.properties API:67
- bindings.xml
 - copying during upgrade AD:344
 - on grid servers AD:50
 - on proxy servers AD:300
 - on share servers AD:59
- build.xml for data service plug-ins P:184, P:197
- cached, permissions on O:50
- caching of O:56
- changing ownership C:42
- changing permissions for AD:239
- clearing locks C:163
- configuring caching P:107
- configuring encryption level C:216
- connectinfo.txt AD:131
- copying AD:213, C:46
- copying locally AD:215
- data type mapping
 - about C:301
 - command to specify location C:68
 - format of C:301
 - deleted, purging from cache C:26
 - deleting AD:219, C:195
 - deleting pin schedules for C:135
 - DGAS properties AD:76
 - dgas_log.xml, DGAS log properties file AD:317
 - displaying C:35
 - evicting from cache P:135
 - for data service plug-ins P:180
 - icon for O:29
 - in the data grid O:8
 - install.exe AD:22, AD:24
 - invalidating all in cache C:28
 - invalidating one in cache C:27, P:136
 - JAR files for data service plug-ins P:180
 - jboss-service.xml, request log properties file AD:328
 - jdbc-log4j.properties API:66
 - join.properties file on proxy servers AD:300
 - krb5.conf AD:152
 - linking AD:217, C:161
 - listing C:166
 - listing locks C:164
 - listing schedules C:136
 - listing those pinned for caching P:134
 - log4j.xml,
 - Avaki Studio log properties file AD:317
 - server log properties file AD:317
 - manifest files for data service plug-ins P:180, P:197
 - marking for no caching C:136, P:132
 - marking for scheduled caching P:120
 - moving AD:210, C:176
 - permissions on new files O:49
 - pinning for scheduled caching C:137
 - properties files for DGAS C:293
 - purging all from cache and unpinning C:25
 - purging from cache and unpinning C:24
 - readme AD:12, AD:15, AD:339
 - refreshing in cache C:32
 - renaming AD:212
 - rendering results into O:25
 - scheduling for caching C:130
 - searching for AD:233
 - shareserver.ports
 - on grid servers AD:50
 - on share servers AD:60
 - system.properties AD:129
 - tagging for on-demand caching C:134, P:129
 - temporary, for sorting large result sets S:76
 - unscheduling cached files P:135
 - uploading to the data catalog AD:282
 - Workbench_project S:11
- fileWrite SOAP operation API:23
- Firefox
 - version requirements AD:5
 - setting for selecting run-as users P:27, P:54, P:92, P:224, P:227
- fonts in Studio display, setting S:23

functions
 in expressions S:73
 used with Aggregate operator S:110

G

garbage collection for views C:273
 GDCs, See grid domain controllers
 generated views
 about O:25, P:217, P:240
 defined AD:354, API:88, C:312, O:66, P:294, S:181
 running P:240
 scheduling updates P:231
 transactional consistency of P:217
 GeneratedViews directory O:33
 generating schemas S:19
 Generator operator S:117
 getAttributes SOAP operation API:23
 getCatalogs method API:75
 getCatalogTerm method API:75
 getDBOpOutput SOAP operation API:54
 getDBOpOutputAttach SOAP operation API:55
 getDBOpParameters SOAP operation API:56
 getDBOpSchema SOAP operation API:56
 getDBOpSchemaAttach SOAP operation API:57
 getDBOpSchemaString SOAP operation API:58
 getDSOutput SOAP operation API:37
 getDSOutputAttach SOAP operation API:38
 getDSOutputString SOAP operation API:38
 getDSParameters SOAP operation API:39
 getOutputString SOAP operation API:55
 getSchemas method API:75
 getSchemaTerm method API:75
 getSQL SOAP operation API:58
 getSystemAttributes SOAP operation API:24
 getUserAttributes SOAP operation API:24
 GIDs, configuring AD:68, C:217, C:221
 Global Parameters menu S:71
 grid directories, See directories, Avaki
 grid domain controllers
 backing up and restoring AD:113
 creating C:127
 defined AD:355, API:89, C:313, O:67, P:295, S:181
 DNS aliases for AD:10
 loading AD:14
 ports used by AD:6, AD:8
 primary AD:355, API:89, C:313, O:67, P:295, S:181
 secondary AD:43
 starting AD:14
 stopping AD:38, C:11
 stopping and restarting GDCs registered as services AD:39
 grid domains
 See domains, Avaki
 grid servers
 associating with caches P:111
 backing up and restoring AD:113
 choosing for web services API:5
 configuring AD:48, AD:52

grid servers (continued)
 configuring associated cache service C:33
 configuring cache service threads AD:131
 configuring nondefault ports AD:50
 connecting C:225
 connection info, setting S:23
 defined AD:355, API:89, C:313, O:67, P:295, S:181
 destroying C:226
 disabling auto-restart on C:12
 disassociating from caches P:112
 displaying associated cache C:27
 displaying status of operations on C:248
 enabling auto-restart C:9
 finding connect ports AD:53
 finding server names AD:53
 installing JDBC drivers on AD:49
 monitoring AD:99
 obtaining upgrade information C:250
 ports used by AD:6, AD:8
 registering C:9
 request logs for AD:327
 server logs AD:317
 settable attributes of C:290
 setting location of temp directory for AD:135
 setting plug-in cache size properties AD:137
 setting up command clients on AD:94
 starting AD:50, C:9, C:10
 stopping AD:51, C:11, C:12, C:227
 stopping and restarting AD:52
 time required to upgrade AD:340
 uncoupling associated cache C:34
 unregistering C:12
 upgrading C:249
 grid user accounts, See users
 grid-server --register command C:9
 grid-server --start command C:10
 grid-server --stop command C:11
 grid-server --unregister command C:12
 group --add --user command C:138
 Group By operator S:76, S:121
 group --create command C:141
 group --delete command C:143
 group --delete --user command C:144
 group --info command C:145
 group --list-user command C:147
 group mappings, adding for a particular DGAS C:92
 groups
 about O:43
 activating privileges for newly added users AD:192, AD:243
 adding to ACLs AD:243, S:97
 adding users to AD:191, C:138
 Administrators AD:45, O:43
 configuring default mappings C:218
 creating AD:191, C:141
 DatabaseAdministrators O:44
 DataProviders O:44
 default grid groups O:43
 default groups for DGAS AD:333

groups (continued)

- defined S:176
- deleting AD:198, C:143
- deleting users from C:144
- displaying information about AD:195, C:145
- enabling interconnection access AD:304
- everyone group O:44
- imported groups O:43
 - from LDAP AD:159
 - from NIS AD:165
 - refreshing AD:195
- in Avaki, defined AD:350, API:84, C:308, O:62, P:290
- listing C:144
- listing users in C:147
- making account changes take effect immediately for DGAS access AD:117
- MessagingUsers O:44
- modifying AD:195
- removing from ACLs AD:242
- removing users from AD:193
- setting up for DGAS AD:67
- UserAdministrators AD:45, O:44
- using in ACLs for cached objects O:50

H

hard links

- about O:36
- broken, to generated views P:240
- creating AD:217, C:161
- defined AD:355, API:89, C:313, O:67, P:295, S:182

hardware requirements for Avaki AD:2

help command C:148

help, online, for command line AD:xiii, C:xiii, P:ix

hideCatalogs JDBC property API:70

home directories AD:169

host names

- aliasing for GDCs AD:10
- setting for servers AD:32

HTTP and HTTPS ports

- default AD:8, AD:9, AD:10
- properties for AD:140

HTTP and web services API:7, API:8

HTTP POST problem in web browsers AD:5

HTTP request logs, See request logs

HTTPS and web services API:7, API:8

hyphens in command syntax AD:xvi, C:xvi, P:xii

I

IATEMPDIR environment variable AD:16

IBM AIX O:16

IBM AIX requirements AD:3

IBM DB2, versions and JDBC drivers for use with Avaki AD:6

icons for grid objects in the data catalog O:29

id command C:149

identity mapping P:6, S:36

imported groups, See groups

imported user accounts AD:167

increment field for Iterator operators S:151

indent size property for XML files AD:142

initialize field for Iterator operators S:151

initParameter element P:263

inner join S:156

input parameters

- creating P:88
- for data services
 - configuring P:84
 - deleting P:86

Input Source element S:125

input sources

- accessing columns from S:68
- browsing for in data catalog view S:18
- configuring for view models in Avaki Studio S:43
- creating S:55
- error handling S:143
- finding S:18

input streams, for data services, configuring P:87

inputParameter element P:264

inputSource element P:265

InputSource interface for data services P:186

InputStream element P:266

installation directory AD:350, API:84, C:308, O:62, P:290, S:176

installing Avaki

- about AD:13
- in Unix AD:16
- in Windows AD:23
- preparation and planning AD:1
- system requirements AD:2

integration, See data integration

interconnection IDs

- creating AD:295, C:149
- using in permissions AD:304, C:43
- using to provide cross-domain data access AD:295

interconnections between grid domains

- about AD:289
- breaking C:127
- creating AD:291
- defined AD:355, API:89, C:313, O:67, P:295, S:182
- disconnecting domains AD:314
- enabling access AD:299
- prerequisites for AD:94
- setting up C:128

two-way, exposing users AD:308

user access methods AD:294

viewing interconnected domains AD:310

Interconnects directory O:33

Internet Explorer

Avaki version requirements AD:5

setting for selecting run-as users P:27, P:54, P:92, P:224, P:227

Intersection operator S:148

performance S:76

IP address, setting for a server AD:32

isDSAvakiXML SOAP operation API:40

isList element P:266
 Iterator operator S:149
 example S:152

J

JAR files

for Avaki JDBC driver API:66
 configuring path for second JAR on one grid server P:247
 for data service plug-ins P:180
 for jConnect API:67

jarurl element P:267

Java

data service plug-in code samples P:190
 sample web services client API:9
 writing data service plug-ins in P:183

Java transformers

error logging P:246
 implementing P:243
 installing P:245
 referring to other documents P:245
 using P:245

java.io.tmpdir system property AD:135

java.protocol.handler.pkgs system property AD:141

java.rmi.server.hostname system property AD:32

java.security.krb5.conf system property AD:143, AD:153

java.security.krb5.kdc system property AD:153

java.security.krb5.realm system property AD:153

java.sql.DatabaseMetaData interface API:75

Javadoc, Avaki, accessing P:185

JavaScript

files S:11
 include files S:11, S:75
 sample S:115
 methods on data service plug-in objects P:201
 resources for learning about S:67
 using Java classes and interfaces in data service plug-ins P:200
 writing data service plug-ins in P:200

JavaScript expressions

about S:4, S:66
 menu for constructing S:71
 uses of in Avaki Studio S:vii

jConnect, See JDBC drivers

JDBC

accessing data catalog through O:24
 and caching of database operation results API:62
 connection properties API:68
 data types API:76
 directing queries to a particular grid server API:70
 result set types API:75

JDBC drivers

about API:61
 Avaki
 choosing version of API:66
 connection properties API:68
 JAR files for API:66

JDBC drivers (continued)

Avaki

setting classpath for API:67
 when to use API:65

batch mode API:77

choosing API:65

configuring for a database connector P:5

configuring two versions on one grid server P:247

connection strings API:71

defined AD:356, API:90, C:314, O:68, P:295, S:182

for supported DBMSes AD:3

installing AD:49

jConnect

changing default port API:71

setting classpath for API:67

using with Sybase databases AD:6

when to use API:65

loading API:68

prerequisites for using API:64

sample code API:77

setting system properties for API:67

supported features API:74

JDBC fetch size

setting for database connectors P:5, S:36

setting for database operations P:254

JDBC schema names API:69

jdbc-log4j.properties file API:66

join algorithms S:155

Join operator S:154

in tutorial S:44

performance S:76

join types S:156

join.properties file on proxy servers AD:300

JRE versions supported by Avaki AD:5

K

keepalive properties for HTTP ports AD:140

Kerberos

configuring with LDAP authentication services AD:152

system properties for AD:143

krb5.conf Kerberos configuration file AD:152

L

last access time file attribute C:290

LBF AD:280

LDAP

authentication services AD:148

See also authentication services, LDAP

authentication through DGAS AD:86

configuring for Kerberos access AD:152

disabling import on login AD:157

host port, default and nondefault AD:149

importing users on login AD:157

specifying a nondefault host port C:158

ldap --add-schedule command C:149

ldap --delete command C:153
 ldap --delete-schedule command C:154
 ldap --import command C:155
 ldap --info command C:157
 ldap --integrate command C:157
 ldap --list-schedules command C:158
 ldap --searchbase command C:160
 ldap --update command C:160
 links
 command for creating C:161
 uses of in data catalog O:36
 See also hard links, soft links
 Linux requirements AD:3
 listDBConns SOAP operation API:59
 listDBOps SOAP operation API:59
 listDBOpsByDBConn SOAP operation API:60
 listDomains SOAP operation API:25
 listDSs SOAP operation API:40
 listSearches SOAP operation API:25
 ln command C:161
 load balancing factor for share servers AD:280, C:244
 local caches AD:356, API:90, C:314, O:14, O:68, P:296, S:182
 locks command C:163
 locks on files
 clearing C:163
 obtaining a list C:164
 log properties file, sample AD:323
 log4j AD:318, P:196
 logging
 audit logs AD:108
 categories of loggable events AD:318
 configuring audit logging AD:319
 for data service plug-ins P:196
 for JDBC API:66
 for TrAX transformers P:246
 for type mapping C:304
 HTTP request logs AD:108
 log4j properties files for servers and Studio AD:317
 properties files for request logs AD:327
 viewing the server log AD:107
 logging in AD:199, C:164
 logical operators S:72
 logical operators in searches AD:234
 logicBox element P:268
 login command C:164
 login info, setting S:23
 logout command C:165
 ls command C:166
 ls SOAP operation API:26
 lsSize SOAP operation API:26

M

manifest files for data service plug-ins P:180, P:197
 mappings
 between Avaki and local users/groups for DGAS AD:68
 between Avaki users and database users, See database identity mappings

mappings (continued)
 database identity
 adding C:252
 deleting C:253
 listing C:255
 default, setting up AD:71
 DGAS default AD:69, C:220
 DGAS domain-wide
 for groups, setting up AD:70
 for users, setting up AD:70
 users, groups, and defaults AD:68
 DGAS dynamic C:109
 for data types in SQL views C:68, C:301
 for users and groups for DGAS C:94, C:109
 per-DGAS
 adding for groups C:92
 adding for users C:94
 deleting C:100
 deleting for groups C:99
 per-DGAS, users and groups AD:88
 self mappings for users C:109
 See also data type mappings
 maximum concurrent data services setting for execution services AD:109
 memory requirements for Avaki software AD:3
 message tests in monitor services AD:101
 message timeout properties for Avaki servers and clients AD:133
 MessagingUsers group and user accounts O:44
 metadata O:13, S:3
 Metadata directory O:33
 metadata models
 about S:77
 creating S:84
 defined AD:356, API:90, C:314, O:68, P:296, S:182
 deleting S:94
 deploying C:285, S:91
 deriving S:92
 editing S:84
 files associated with S:11
 importing S:79
 mapping to Avaki objects S:88
 naming scheme S:91
 undeploying C:286, S:94
 Microsoft SQL Server, versions and JDBC drivers for use with Avaki AD:6
 Microsoft Visual Studio API:5
 MicroSoft Windows O:16
 MIME in Avaki web services API:3
 minus signs in command syntax AD:xvi, C:xvi, P:xii
 mkdir command C:167
 mkdir SOAP operation API:27
 mkdirParents SOAP operation API:27
 mkdirServer SOAP operation API:29
 mkdorParentsServer SOAP operation API:28
 models, See metadata models and view models
 modification time file attribute C:290

monitor --add command C:167
 monitor --clear command C:168
 monitor --create command C:170
 monitor --delete command C:171
 monitor --list command C:172
 monitor --result command C:172
 monitor services
 monitor --start command C:173
 monitor --stop command C:174
 monitoring
 about AD:99
 adding tests C:167
 configuring AD:101
 creating monitor services C:170
 deleting monitor services AD:106, C:171
 deleting tests AD:105
 disabling and enabling tests AD:104
 listing active tests C:172
 logging AD:107
 message tests AD:101
 ping tests AD:100
 removing tests C:168
 restarting tests AD:105, C:173
 stopping tests AD:105, C:174
 viewing results AD:103, C:172
 mount port for DGAS AD:81
 mount protocol port, default AD:9
 Mozilla, Avaki version requirements AD:5
 Multiplexer operator S:157
 multiplexing socket properties AD:142
 mv command C:176
 mv SOAP operation API:29
 MySQL
 configuring XA driver for P:7, S:36
 versions and JDBC drivers for use with Avaki AD:6

N

name element P:269
 name property for columns S:61
 names of Avaki objects
 about O:29
 avoiding underscores in when using BusinessObjects API:81
 case sensitivity and restrictions AD:206
 changing AD:212
 metadata models and mapped tables S:91
 of elements S:58
 qualified names O:30
 restrictions on AD:41
 three-part O:24
 navigator in Studio S:17
 nesting operations in data services S:149
 .NET, See .NET under Symbols at the beginning of the index
 Netscape requirements AD:5
 NFS
 and permissions AD:12
 configuring NFS port for DGAS AD:81

NFS (*continued*)
 port, default AD:9
 shutting down before starting a DGAS AD:65
 NFS clients
 attribute caching for AD:336
 defined AD:356, API:90, C:314, O:68, P:296, S:183
 older, accessing data grid through AD:337
 requirements for O:16
 setting up AD:91
 NFS URLs AD:93
 NIS
 disabling import on login AD:163
 importing users on login AD:163
 See also authentication services, NIS
 nis --add-schedule command C:177
 nis --delete command C:181
 nis --delete-schedule command C:181
 nis --import command C:182
 nis --info command C:183
 nis --integrate command C:184
 nis --list-schedules command C:184
 nis --update command C:185
 NLM AD:64
 NLM protocol port, default AD:9
 notifications, See update notifications

O

object host name grid server attribute C:290
 octothorpe AD:33, AD:129
 ODBC
 accessing data catalog through O:24
 defined AD:356, API:90, C:314, O:68, P:296, S:183
 support for API:80
 ODBC drivers, using with Avaki API:80
 offline expiration cache attribute C:291
 on-demand caching
 about P:119
 defined AD:357, API:91, C:315, O:69, P:296, S:183
 of database operation and data service results O:57, P:108
 of files O:56, P:107
 online help for command line AD:xiii, C:xiii, P:ix
 operating systems supported by Avaki O:16
 operations, monitoring AD:99
 operators
 about S:5
 adding to a view model S:55
 Aggregate S:108
 connecting S:57
 Custom S:111
 descriptions of S:59
 Generator S:117
 Group By S:121
 in searches AD:234
 Input Source S:125
 Intersection S:148
 Iterator S:149
 Join S:154

- operators (*continued*)
 - logical, in expressions S:72
 - moving S:56
 - Multiplexer S:157
 - names of S:58
 - Order By S:159
 - performance considerations S:5
 - Projection S:161
 - properties dialogs S:58
 - Result S:164
 - Select S:165
 - selecting S:56
 - sort-based, performance of S:76
 - Splitter S:166
 - Union S:168
 - Update S:169
 - with red borders S:60
 - Oracle 10g
 - configuring XA driver for P:7, S:36
 - versions and JDBC drivers for use with Avaki AD:6
 - Order By operator S:159
 - performance S:76
 - os arch grid server attribute C:290
 - os name grid server attribute C:290
 - outer-full join S:156
 - outer-left join S:156
 - outer-right join S:156
 - output streams, for data services, configuring P:86
 - outputStream element P:269
 - ownership of objects in the data catalog
 - about O:46
 - setting AD:242
- P**
- palette in Avaki Studio view model editor S:20, S:54
 - parameters
 - accessing in expressions S:71
 - adding S:64
 - deleting S:65
 - displaying S:63
 - for data service plug-ins
 - about P:178
 - specifying for Java plug-ins C:188
 - specifying for XSLT plug-ins P:181
 - for testing view models S:50
 - in Avaki Studio, about S:4
 - input, for data services, configuring P:84
 - mapping input parameters S:144
 - modifying S:65
 - reordering S:65
 - validating S:64
 - ParameterSpec interface for data services P:187
 - passwords
 - changing AD:175, C:185
 - specifying for JDBC connections API:69
 - patches, settable attributes of C:290
 - performance S:5, S:76
 - benefits of caching O:54
 - tracking, enabling in an Avaki shell C:245
 - Perl, sample web services client API:9
 - permissions
 - about O:45
 - changing AD:239, C:41
 - granted by grid groups O:43
 - hiding objects with O:49
 - on new Avaki shares AD:261
 - on shared data AD:12
 - setting in ACLs S:97
 - values for AD:242, O:47
 - viewing for Avaki services C:186
 - See also ACLs
 - permissions command C:186
 - permissions SOAP operation API:30
 - perspectives in Avaki Studio
 - Avaki S:13
 - defined S:12
 - Resource S:11
 - pin for caching, defined AD:357, API:91, C:315, O:69, P:297, S:183
 - ping tests in monitor services AD:100
 - planning an Avaki deployment AD:1
 - platforms supported by Avaki O:16
 - plugin command P:184
 - plugin --generate command C:187
 - Plugin interface for data services P:186
 - Plug-in Wizard and creating data service plug-ins P:183
 - plug-ins, See data service plug-ins
 - plus signs in command syntax AD:xvi, C:xv, P:xi
 - ports
 - bad port cache AD:141
 - changing, See ports, nondefault
 - CIFS, releasing before running a DGAS AD:66
 - default, for Avaki servers AD:6
 - HTTP and HTTPS, See HTTP and HTTPS ports
 - LDAP host
 - default and nondefault AD:149
 - specifying C:158
 - NFS, default AD:9
 - nondefault
 - configuring for DGAS AD:74, AD:75
 - configuring for grid servers AD:50
 - configuring for proxy servers AD:300
 - configuring for share servers AD:59
 - RMI, See RMI ports
 - SMB, default AD:9
 - specifying in WSDL API:6
 - SSL, See SSL ports
 - TDS AD:8, AD:50, AD:145, API:71
 - ports, connect, See connect ports
 - POST problem in web browsers AD:5
 - precision property for columns S:61
 - preferences for Avaki Studio, setting S:23
 - primary GDCs AD:357, C:315, O:69

- privacy and HTTPS API:8
- Projection operator S:161
 - in tutorial S:46
- projects
 - creating S:13
 - defined S:12
- properties
 - cache sizes for data service plug-ins AD:137
 - DGAS
 - configuring AD:82
 - controlling cache size AD:124
 - displaying AD:124, C:103
 - displaying descriptions C:104
 - listing C:103
 - properties file C:293
 - setting C:113
 - unsetting C:117
 - for cache services AD:135
 - for encryption of grid objects AD:139
 - for HTTP and HTTPS ports AD:140
 - for HTTP keepalives on proxy servers AD:140
 - for JDBC clients AD:128
 - for JDBC connections API:68
 - for Kerberos AD:143
 - for multiplexing sockets AD:142
 - for remote object stub cache AD:144
 - for schedule exclusion cache AD:144
 - for server request logs AD:327
 - for servers acting as clients AD:128
 - for share servers AD:138
 - for virtual database table cache size AD:144
 - for XA connections C:70, P:7, S:37
 - Java system properties, providing to JDBC driver API:67
 - JDBC, specifying for a database connector C:67
 - Kerberos default realm AD:153
 - Kerberos key distribution center AD:153
 - location of Kerberos configuration file AD:153
 - message timeouts for Avaki servers and clients AD:133
 - remote coherence window for configurations AD:141
 - setting server's host name or IP address AD:33
 - system. See system properties.
 - views
 - displaying C:274
 - setting C:275
 - XML indent size AD:142
- properties dialog boxes S:58
- provisioning data O:21
- proxy --add command C:191
- proxy --delete command C:191
- proxy --list command C:192
- proxy routing tables
 - about AD:289
 - configuring AD:292, C:191
 - displaying C:192

- proxy servers
 - about AD:289
 - configuring AD:299
 - configuring nondefault ports AD:300
 - connecting C:228
 - defined AD:357, API:91, C:315, O:69, P:297, S:183
 - deleting from the routing table C:191
 - destroying C:228
 - disabling auto-restart C:14
 - enabling auto-restart AD:300, C:12
 - finding connect port numbers AD:303
 - finding server names AD:303
 - installing in Windows AD:24
 - installing in Unix AD:18
 - ports used by AD:10
 - registering for auto-restart AD:302, C:12
 - request logs for AD:327
 - server logs for AD:317
 - setting HTTP keepalive properties for AD:140
 - setting up C:191
 - starting AD:301, C:12, C:13
 - stopping C:14, C:228
 - stopping and restarting AD:301
 - time required to upgrade AD:341
 - unregistering C:14
 - when to deploy AD:290
- proxy-server --register command AD:302, C:12
- proxy-server --start command C:13
- proxy-server --stop command C:14
- proxy-server --unregister command C:14
- pwd command C:193

Q

- qualified names
 - about O:30
 - for data services, specifying API:35, C:51, C:267
 - for database connectors, specifying C:254
 - for database operations, specifying C:25, C:264
 - for groups, specifying C:43, C:142, C:145, C:147, C:148
 - for users, specifying C:42, C:53, C:60, C:86, C:89, C:141, C:145, C:165, C:251, C:266, C:280
- queries, See ad-hoc queries
- query engine
 - defined AD:357, API:91, C:315, O:69, P:297, S:184
 - mapping data types for C:301
 - sort chunk size property AD:139
- queryCacheTTL JDBC property API:71

R

- RAM requirements for Avaki software AD:3
- range input for Iterator operators S:151
- readme file AD:12, AD:15, AD:339
- red borders on operators in Studio S:60
- Red Hat Linux requirements AD:3
- ref element P:270

refresh schedules
 for Avaki shares AD:266, C:231
 advanced AD:272
 calendared AD:271
 exclusions for AD:274
 listing C:239
 one-time AD:269
 periodic AD:270
 removing AD:278, C:236
 for data services P:152
 advanced P:157
 calendared P:155
 one-time P:155
 periodic P:157
 for database operations P:139
 advanced P:144
 calendared P:142
 one-time P:142
 periodic P:144
 for files or directories P:120
 advanced P:127
 calendared P:125
 one-time P:125
 periodic P:127
 for generated views P:231
 advanced P:236
 calendared P:234
 one-time P:233
 periodic P:236
 for LDAP authentication services AD:185
 for virtual database operations P:139
 advanced P:144
 calendared P:142
 one-time P:142
 periodic P:144
 refreshing users on login AD:149
 reindex interval for search services AD:232
 remote caches AD:357, API:91, C:315, O:14, O:69, P:297,
 S:184
 removeAttribute SOAP operation API:31
 replica --add command C:193
 replica --config command C:193
 replica --delete command C:194
 replica --info command C:194
 replica --synch command C:195
 request logs
 configuring AD:327
 viewing AD:108
 requirements, pre-installation AD:2
 Result element S:164
 result sets
 accessor functions S:73
 combining S:44
 large, providing space for sorting S:76
 types in JDBC API:75

rm command C:195
 rm SOAP operation API:31
 RMI ports
 default AD:8, AD:9, AD:10
 linked to grid server connect ports AD:50
 linked to share server connect ports AD:60
 routing tables, configuring AD:292, C:191
 row-level access control S:74
 RowSetFactory class for data services P:188
 rowsets O:11, P:273
 rpc/encoded web services API:3, API:5
 rpcinfo command AD:66
 Rudi port AD:352, C:310, O:64
 run-as users, See users and user accounts

S

Saxon C:265, C:268, C:271, P:220, P:224, P:227
 scale property for columns S:61
 schedule --delete command C:196
 schedule exclusions
 about AD:274, P:166
 adding for Avaki share rehashes C:234
 adding for data services C:51
 adding for directories C:125
 adding for files C:133, C:140
 adding for LDAP authentication services C:152
 adding for NIS authentication services C:180
 adding for views C:262
 adding to Avaki directories C:121
 adding to database operations C:76
 applying to schedule entries P:168
 caching properties for AD:144
 configuring AD:274, P:166
 creating custom C:198
 creating daily C:199
 creating monthly C:201
 creating weekly C:203
 creating yearly C:205
 defined AD:358, API:92, C:316, O:70, P:298, S:184
 deleting C:207
 displaying information about C:208, P:171
 listing names C:209
 schedule --info command C:197
 schedule --list command C:197
 schedule --print-iterations command C:198
 scheduled caching
 about P:119
 defined AD:357, API:91, C:315, O:69, P:297, S:184
 of database operation and data service results O:59, P:109
 of files O:57, P:107
 scheduleexclusion --create --custom command C:198
 scheduleexclusion --create --daily command C:199
 scheduleexclusion --create --monthly command C:201
 scheduleexclusion --create --weekly command C:203
 scheduleexclusion --create --yearly command C:205
 scheduleexclusion --delete command C:207
 scheduleexclusion --info command C:208

- scheduleexclusion --list command C:209
- schedules
 - adding for data services C:48, P:152
 - adding for database operations C:73, P:139
 - adding for directories C:117
 - adding for files C:130
 - adding for LDAP user importation C:149
 - adding for NIS user importation C:177
 - adding for views C:259
 - adding for virtual database operations P:139
 - creating cron specifications C:297
 - creating custom exclusions C:198
 - creating daily exclusions C:199
 - creating monthly exclusions C:201
 - creating weekly exclusions C:203
 - creating yearly exclusions C:205
 - cron expressions in AD:185, AD:273
 - deleting C:196
 - deleting exclusions C:207
 - deleting for data services C:54
 - deleting for database operations C:78
 - deleting for directories C:122
 - deleting for files C:135
 - deleting for LDAP user importation C:154
 - deleting for NIS user importation C:181
 - deleting for views C:272
 - displaying exclusion information C:208
 - displaying information about C:197
 - exclusions, see also schedule exclusions
 - execute permissions required O:59
 - for refreshing LDAP authentication services AD:185
 - for view generators and generated views P:231
 - listing C:197
 - listing execution times C:198
 - listing for data services C:58
 - listing for database operations C:88
 - listing for directories C:126
 - listing for files C:136
 - listing for LDAP authentication services C:158
 - listing for NIS authentication services C:184
 - listing for views C:274
 - listing names of exclusions C:209
 - refresh, See refresh schedules
 - types of AD:185, AD:267
- schemas
 - about S:3, S:60
 - enabling browsing on a database connector P:4
 - expressed in metadata models S:77
 - for Avaki data services API:75
 - for operators, column properties of S:60
 - for virtual database operations and their SQL views API:75
 - for virtual database operations, generating P:57
 - generating S:19
 - generating for data services C:55, P:98
 - generating for database operations C:80, P:31
 - getting information about through JDBC API:75
 - getting via JDBC API:75
- schemas (*continued*)
 - JDBC schema names API:69
 - modifying S:161
 - types in Avaki domains P:51, P:64
 - viewing for databases P:9
 - See also metadata models
 - search (execute) command C:211
 - search --create command C:209
 - search --delete command C:210
 - search --get-rehash-level command C:212
 - search --info command C:214
 - search --rehash command C:215
 - search --set-rehash-level command C:215
 - search SOAP operation API:32
 - searches O:13
 - configuring rehash intervals C:215
 - creating search services AD:231, C:209
 - deleting AD:236
 - deleting search services C:210
 - displaying search service information C:214
 - performing AD:233, C:211
 - rehashing search services C:215
 - reindex interval, setting AD:232
 - viewing rehash intervals C:212
 - SearchQuery complex type API:17
 - SearchResult complex type API:17
 - secondary GDCs AD:43
 - adding C:193
 - deleting C:194
 - forcing updates C:195
 - setting refresh intervals C:193
 - setting update interval C:195
 - viewing C:194
 - See also grid domain controllers
 - security
 - .NET API:9
 - about O:10
 - ACLs O:45
 - authentication O:41
 - configuring encryption levels C:216
 - displaying encryption levels C:222
 - for web clients API:8
 - HTTPS API:8
 - setting permissions C:41
 - SSL certificates API:8
 - user accounts and groups O:43
 - viewing permissions C:186
 - security --config command C:216
 - security --default-gid command C:217
 - security --default-group command C:218
 - security --default-uid command C:219
 - security --default-user command C:220
 - security --gid command C:221
 - security --info command C:222
 - security --uid command C:222
 - Select operator S:165
 - server connect ports, See connect ports
 - server --dgas --connect command C:223

- server --dgas --destroy command C:224
- server --dgas --stop command C:225
- server --grid --connect command C:225
- server --grid --destroy command C:226
- server --grid --stop command C:227
- server logs
 - configuring AD:317
 - viewing AD:107
- server --proxy command C:228
- server --share --connect command C:228
- server --share --disconnect command C:229
- server --share --stop command C:230
- servers, backing up databases for C:23
- servers, Avaki
 - defined AD:350, API:84, C:308, O:62, P:290, S:176
 - displaying software version of AD:99
 - finding names of AD:131
 - in a typical deployment O:18
 - monitoring AD:99
 - ports used by AD:6
 - problems communicating with AD:32
 - setting cache service properties for AD:135
 - setting host name or IP address to advertise AD:32
 - setting message timeout properties for AD:133
 - See also DGAS, grid domain controllers, grid servers, share servers, and proxy servers
- servers, proxy, See proxy servers
- services, Avaki
 - copying C:46
 - defined AD:358, API:92, C:316, O:70, P:298, S:184
 - icon for O:29
- setAttribute SOAP operation API:32
- setOutputStream JavaScript method for data service plugins P:202
- share --add-rehash-schedule command C:231
- share --add-share-servers command C:232
- share --create command C:235
- share --delete-rehash-schedule command C:236
- share --disconnect command C:238
- share --get-local-path command C:238
- share --get-status command C:239
- share --list-rehash-schedules command C:239
- share --list-share-servers command C:240
- share --rehash command C:240
- share --remove-share-servers command C:241
- share servers
 - about AD:54
 - adding to Avaki shares AD:263
 - backing up and restoring AD:115
 - before disconnecting AD:54, C:230
 - configuring a machine with one share server AD:55
 - configuring multiple share servers on one machine AD:59
 - configuring to use nondefault ports AD:59
 - connecting to grid servers C:228
 - defined AD:358, API:92, C:316, O:70, P:298, S:184
 - disabling auto-restart C:17
 - disconnecting from grid servers C:229
 - enabling auto-restart C:15
 - finding connect ports AD:59
 - finding server names AD:59
 - installing in Windows AD:24
 - installing on Unix AD:18
 - local path AD:261
 - modifying load balance factor C:244
 - multiple AD:55, AD:263
 - obtaining upgrade information C:250
 - ports used by AD:9
 - registering C:15
 - registering for auto-restart AD:57
 - removing from Avaki shares C:241
 - removing from shares AD:265
 - replacing for Avaki shares C:242
 - server logs for AD:317
 - setting for Avaki shares AD:260
 - setting load balancing factor AD:280
 - setting system properties for AD:138
 - starting AD:56, C:15, C:16
 - stopping C:17, C:230
 - stopping and restarting AD:57
 - time required to upgrade AD:341
 - unregistering C:17
 - upgrading C:249
 - write access and user accounts AD:12
- share --set-local-path command C:241
- share --set-share-servers command C:242
- share --set-status command C:243
- share --update-share-servers command C:244
- shared directories, See Avaki shares
- shares, See Avaki shares and CIFS shares
- Shares directory O:34
- shares, CIFS AD:125
- share-server --register command AD:57, C:15
- share-server --start command C:16
- share-server --stop command C:17
- share-server --unregister command C:17
- shareserver.ports file
 - on grid servers AD:50
 - on share servers AD:60
- shell command C:245
- shells, Avaki, accessing C:245
- shortcuts created in Windows installations AD:27
- SMB ports, default AD:9
- SOAP
 - formal definition API:1
 - learning about API:1
 - over HTTP API:8
 - over HTTPS API:8
 - standards compliance API:1
- SOAP clients, See web services clients
- SOAP complex types API:12
 - AdHocDBOPExecutionParams API:13
 - AvakiPrincipal API:13
 - DataCatalogAttribute API:14
- share servers (*continued*)
 - enabling auto-restart C:15
 - finding connect ports AD:59
 - finding server names AD:59
 - installing in Windows AD:24
 - installing on Unix AD:18
 - local path AD:261
 - modifying load balance factor C:244
 - multiple AD:55, AD:263
 - obtaining upgrade information C:250
 - ports used by AD:9
 - registering C:15
 - registering for auto-restart AD:57
 - removing from Avaki shares C:241
 - removing from shares AD:265
 - replacing for Avaki shares C:242
 - server logs for AD:317
 - setting for Avaki shares AD:260
 - setting load balancing factor AD:280
 - setting system properties for AD:138
 - starting AD:56, C:15, C:16
 - stopping C:17, C:230
 - stopping and restarting AD:57
 - time required to upgrade AD:341
 - unregistering C:17
 - upgrading C:249
 - write access and user accounts AD:12

SOAP complex types (*continued*)

DataCatalogEntry API:15
 DataCatalogPermission API:15
 DataServiceExecutionParams API:16
 DBOPExecutionParams API:16
 SearchQuery API:17
 SearchResult API:17

SOAP operations

accessibleDBOp API:42
 accessibleDS API:36
 accessiblePath API:19
 chmod API:19
 chown API:20
 components of web service API:4
 data catalog API:18
 data services API:34
 database operations API:40
 executeAdHocDBOp API:43
 executeAdHocDBOpWithoutOutput API:44
 executeAdHocDBOpWithoutOutputAttach API:46
 executeAdHocDBOpWithoutOutputString API:47
 executeDBOp API:48
 executeDBOpBytesInput API:49
 executeDBOpGridFileInput API:50
 executeDBOpWithOutput API:50
 executeDBOpWithOutputAttach API:52
 executeDBOpWithOutputString API:53
 executeDS API:36
 fileRead API:21
 fileReadAttach API:21
 fileReadString API:22
 fileWrite API:23
 getAttributes API:23
 getDBOpOutput API:54
 getDBOpOutputAttach API:55
 getDBOpParameters API:56
 getDBOpSchema API:56
 getDBOpSchemaAttach API:57
 getDBOpSchemaString API:58
 getDSOutput API:37
 getDSOutputAttach API:38
 getDSOutputString API:38
 getDSParameters API:39
 getOutputString API:55
 getSQL API:58
 getSystemAttributes API:24
 getUserAttributes API:24
 isDSAvakiXML API:40
 listDBConns API:59
 listDBOps API:59
 listDBOpsByDBConn API:60
 listDomains API:25
 listDSs API:40
 listSearches API:25
 ls API:26
 lsSize API:26
 mkdir API:27
 mkdirParents API:27

SOAP operations (*continued*)

mkdirParentsServer API:28
 mkdirServer API:29
 mv API:29
 permissions API:30
 removeAttribute API:31
 rm API:31
 search API:32
 setAttribute API:32
 tester API:33
 whoami API:33
 SOAP::Lite API:5
 sockets, multiplexing AD:142
 setting maximum channels AD:142
 setting maximum write AD:142
 setting send buffer size for AD:143
 soft links
 about O:36
 creating AD:217, C:161
 defined AD:358, API:92, C:316, O:70, P:298, S:185
 not used in searches AD:233
 software requirements for Avaki AD:2
 Solaris requirements AD:3
 sort chunk size
 controlling S:76
 for query engine AD:139
 spaces
 in Windows install pathnames, avoiding AD:26
 to separate arguments in Avaki commands AD:xvi, C:xvi,
 P:xii
 SPARC/Solaris requirements AD:3
 Splitter operator S:166
 SQL
 aggregate functions and aliasing columns S:42
 as prerequisite for Avaki Studio users S:vii
 statements
 in database operations C:86, P:251
 in virtual database operations C:90
 SQL Server, versions and JDBC drivers for use with
 Avaki AD:6
 sql view --delete command C:246
 sql view --get-description command C:246
 sql view --set-description command C:247
 SQL views
 about O:8, O:22, P:38
 adding to categories P:47
 configuring attributes P:44
 data type mappings for C:68, C:301, P:39
 defined AD:358, API:92, C:316, O:70, P:298, S:185
 deleting C:246
 displaying descriptions C:246
 displaying tables provisioned from database connectors C:73
 enabling provisioning on a database connector P:4
 from data service results, generating P:60, P:100
 generated from database operations, removing P:35
 generating from data services C:56
 generating from database operations C:82, P:34
 location in categories S:18

SQL views (*continued*)

- managing P:20
- modifying P:43
- modifying descriptions C:247
- names in data catalog O:24
- provisioning P:39
- provisioning from database connectors C:71
- qualified names for O:31
- removing P:44
- schema types for P:51, P:64
- schemas for S:22
- searching for AD:233
- table types for API:75
- viewing P:42
- viewing and modifying ACLs P:46

square brackets in command syntax AD:xv, C:xv, P:xi

SSL certificates

- about API:8
- generating AD:33
- installing AD:39
- planning for AD:8

SSL ports, default AD:8, AD:9, AD:10

status command C:248

status of grid operations, monitoring AD:99

stored procedures API:73

StreamingRowSet interface for data services P:187

streams, closing in data service plug-ins P:186

Studio, *See* Avaki Studio

style sheet engines

- for data service view generators P:227
- for database operation view generators P:224
- for file view generators P:220

style sheet engines for database view generators C:265, C:268

style sheet engines for file view generators C:271

Sun JDK for compiling data service plug-ins P:184

Sun ONE Directory Server AD:148

- See also* authentication services, LDAP AD:148

Sun Solaris requirements AD:3

SuSE Linux requirements AD:3

Sybase ASA, versions and JDBC drivers for use with Avaki AD:6

Sybase ASE

- configuring XA driver for P:7, S:36
- connection property required for API:70
- versions and JDBC drivers for use with Avaki AD:7

Sybase contact information AD:xvi, API:vii, C:xvi, O:vi, P:xii, S:ix

Sybase IQ, versions and JDBC drivers for use with Avaki AD:7

syntax conventions for commands AD:xv, C:xiv, P:x

system attributes AD:248, S:101

System directory O:34

system properties

- about AD:128
- descriptions of AD:131
- setting on Avaki Studio AD:129
- setting on clients AD:129
- setting on servers AD:129
- setting with JDBC driver API:67

- system requirements for Avaki software AD:2, O:16
- system.properties file AD:33, AD:129

T

table schema view S:22

table types for SQL views API:75

tables

- deleting SQL views C:246
- displaying descriptions C:246
- generating from data services C:56
- generating from database operations C:82
- in metadata models S:77
 - arranging in editor S:84
 - making accessible via JDBC S:91
 - mapping to Avaki objects S:88
 - naming scheme for S:91
- in virtual database, displaying C:283
- mapping data types for C:301
- modifying descriptions C:247
- provisioned from database connectors, displaying C:73
- provisioning as SQL views O:22
- provisioning from database connectors C:71
- qualified names for O:32
- schemas for, displaying S:22
- See also* SQL views

tabs for view models S:21

target element P:270

TCP channel sockets AD:142

TDS port

- changing AD:50, AD:145, API:71
- default AD:8

technical support contact information AD:xvi, API:vii, C:xvi, O:vi, P:xii, S:ix

temp directories for grid servers, setting AD:135

Templates class P:244

tester SOAP operation API:33

testing an upgraded grid domain AD:346

testing function for WS clients API:33

text conventions C:xiv

timeout properties for Avaki server communications AD:133

timeouts, configuring for database operations P:253

transactions, *See* distributed transactions P:78

TrAX API P:243

two-phase commit protocol P:79

type element P:270

type mapping, *See* data types, mapping

type property for columns S:61

TypeMapping log4j category C:304

types for variables S:70

U

- UID attribute in LDAP authentication services AD:150
- UIDs, configuring AD:68, C:219, C:222
- underscore characters in Avaki names API:81
- Union operator S:168
- Unix file mode semantics, setting AD:337

- Unset permission in ACL AD:242, O:47
- update intervals for GDCs, setting C:195
- update notifications
 - configuring P:238
 - defined AD:359, API:93, C:317, O:71, P:299, S:185
 - enabling AD:311
- Update operator S:169
- upgrade command C:249
- upgrade --info command C:250
- upgrading Avaki software
 - interoperability of different versions AD:339
 - preparation steps AD:341
 - upgrade planning AD:340
 - upgrade steps AD:342
- upstream, defined S:3
- urlLogicBox element P:271
- user attributes AD:248, S:101
- user command C:250
- user --create command C:251
- user --db-mapping --add command C:252
- user --db-mapping --delete command C:253
- user --db-mapping --list command C:255
- user --delete command C:257
- user groups, See groups
- user --info command C:258
- user --list-group command C:258
- UserAdministrators group AD:45, O:44
- users and user accounts
 - about O:43
 - adding database identity mappings C:252
 - adding to ACLs AD:243, S:97
 - adding to groups AD:191, C:138
 - administrative accounts, setting up AD:44
 - changing passwords AD:175, C:185
 - clearing credentials from DGAS cache AD:117
 - configuring associated cache service C:33
 - configuring default mappings C:220
 - configuring dynamic mappings C:109
 - configuring self mappings C:109
 - creating accounts AD:168, C:251
 - creating home directories AD:169
 - default users for DGAS AD:333
 - deleting accounts AD:189, C:257
 - deleting database identity mappings C:253
 - deleting from groups C:144
 - disabling import on login (LDAP) AD:157
 - disabling import on login (NIS) AD:163
 - displaying and changing account information AD:187
 - displaying associated cache C:27
 - displaying full names and contact information C:258
 - displaying names C:286
 - enabling interconnection access AD:304
 - enabling on authentication services C:250
 - exposing in a two-way interconnect AD:308
 - giving access to other domains AD:294
 - grid accounts AD:167
 - imported accounts AD:167
 - users and user accounts (*continued*)
 - importing from LDAP services AD:157
 - importing from NIS services AD:164
 - importing on login (LDAP) AD:157
 - importing on login (NIS) AD:163
 - listing database identity mappings C:255
 - listing group membership for C:258
 - logging in AD:199
 - logging out when newly added to groups AD:192, AD:243
 - making account changes take effect immediately for DGAS access AD:117
 - managing AD:167
 - mapping Avaki users to database users, See database identity mappings
 - MessagingUser O:44
 - qualified names for O:32
 - querying whether enabled in LDAP AD:151
 - refreshing imported accounts AD:185
 - refreshing on login (LDAP) AD:149
 - removing from ACLs AD:242
 - removing from groups AD:193
 - roles for O:43
 - run-as users
 - browser setting for selecting P:27, P:54, P:92, P:224, P:227
 - for data service views P:227
 - for data services P:92
 - for database operation views P:223
 - for database operations P:26
 - for virtual database operations P:54
 - setting run-as user for views C:279
 - setting up for DGAS AD:67
 - setting up local accounts for Avaki AD:11
 - specifying for JDBC connections API:69
 - uncoupling associated cache C:34

V

- validation error expressions S:65
- validation expressions S:64
- value element P:272
- values element P:272
- variables
 - about S:69
 - allowed types for S:70
 - downstream, menu of S:71
 - in Avaki Studio, about S:4
 - updating S:69
- VB .NET, See .NET
- versions
 - of Avaki software, displaying AD:99, C:148, C:250
 - SOAP API:1
 - TrAX P:243
 - WSDL API:1
- vertical bars in command syntax AD:xv, C:xv, P:xi
- view --add-schedule command C:259
- view --create --database command C:263
- view --create --data-service command C:266

- view --create --file command C:267
- view --delete command C:272
- view --delete-schedule command C:272
- view --depends command C:272
- view --garbage-collect command C:273
- view generators
 - about O:8, O:25, P:217
 - caching of input files P:241
 - configuring update notifications for P:238
 - defined AD:359, API:93, C:317, O:71, P:299, S:185
 - for data services
 - setting up P:225
 - specifying a style sheet engine P:227
 - for database operations
 - setting up P:221
 - specifying a style sheet engine P:224
 - for files
 - setting up P:218
 - specifying a style sheet engine P:220
 - for large data sets and unsupported formats P:242
 - listing dependent operations P:228
 - modifying P:229
 - non-XSLT-based P:242
 - removing P:239
 - rowsets as inputs of P:275
 - running P:240
 - scheduling updates P:231
 - troubleshooting P:240
 - using TrAX transformers P:242
- view --info command C:274
- view --list-schedules command C:274
- view models
 - about O:23, S:2
 - configuring input sources S:43
 - creating S:42
 - defined AD:359, API:93, C:317, O:71, P:299, S:185
 - deploying as data services S:50
 - error handling S:143
 - files associated with S:11
 - opening, saving and closing S:17
 - red borders showing errors S:60
 - sample workflow for S:29
 - schemas S:3
 - tabs for, in a project S:21
 - testing S:49
 - view model editor S:20
- view --regenerate command C:273
- view --set-property command C:275
- view --update command C:279
- ViewLibrary category, contents of S:18
- views
 - adding generation schedules C:259
 - adding schedule exclusions C:262
 - configuring values for SQL parameters C:275
 - configuring with database sources C:263
 - configuring with data-service sources C:266
 - configuring with file sources C:267
- views (*continued*)
 - deleting C:272
 - deleting generation schedules C:272
 - listing generation schedules for C:274
 - obtaining information about C:274
 - regenerating C:273
 - removing old results C:273
 - setting run-as user C:279
 - showing dependencies C:272
 - specifying a style sheet engine C:265, C:268, C:271
 - updating C:279
 - See also generated views, SQL views, view generators, view models
- virtual database
 - about O:22
 - configuring attributes P:70
 - defined AD:359, API:93, C:317, O:71, P:299, S:185
 - displaying SQL views C:283
 - executing ad-hoc queries on C:282
 - schema
 - browsing catalogs P:64
 - browsing schemas P:64
 - browsing tables P:64
 - table cache size system property AD:144
 - types of schemas P:51, P:64
 - viewing and modifying ACLs P:72
- virtual database operations
 - about O:23, P:49
 - access permissions P:50
 - allowing creation of C:280
 - allowing groups to create P:67
 - allowing users to create P:65
 - creating C:87, P:50
 - defined AD:359, API:93, C:317, O:71, P:299, S:186
 - evicting from cache P:150
 - executing P:61
 - generating SQL views from P:60
 - listing in cache P:148
 - location in categories S:18
 - managing P:50
 - marking for scheduled caching P:139
 - modifying P:55
 - preventing creation of C:281
 - preventing groups from creating P:69
 - preventing users from creating P:68
 - qualified names for O:31
 - removing P:63
 - schemas for, generating P:57
 - SQL statements in C:90
 - tagging for on-demand caching P:146
 - unscheduling P:150
 - viewing P:55, P:57, P:59
 - viewing dependencies P:59
 - viewing details P:56
- virtual database service, configuring access permissions P:65
- virtualdatabase --allow-dbp-creation command C:280
- virtualdatabase --disallow-dbp-creation command C:281
- virtualdatabase --execute command C:282

virtualdatabase --show-tables command C:283
 virtualeschema --deploy command C:285
 virtualeschema --undeploy command C:286
 virtual schema models, See metadata models

W

web browsers

requirements for Avaki software AD:5
 setting for selecting run-as users P:27, P:54, P:92, P:224,
 P:227

web services API:2, API:8

about API:2
 access permissions API:9
 client examples API:9
 data catalog API:18
 data services API:34
 database operations API:40
 development framework API:5
 document/literal API:3, API:5
 provisioning, about P:205
 rpc/encoded API:3, API:5
 security API:8
 with MIME API:3

web services clients

defined AD:359, API:93, C:317, O:71, P:299, S:186
 requirements for API:4

web services description language, See WSDLs

whoami command C:286

whoami SOAP operation API:33

wildcard characters in searches AD:235

Windows

avoiding install pathnames with spaces AD:26
 installing Avaki in AD:23
 installing on Windows 2003 AD:22
 requirements for AD:3
 services, running under avaki local user account AD:11
 shortcuts for AD:27
 update for HTTP POST problem in web browsers AD:5
 versions supported by Avaki O:16

Windows domains

displaying for DGAS admission policies C:114
 setting for DGAS admission policies C:111
 unsetting for DGAS admission policies C:116

Windows Services list C:5

Workbench S:12

workspace directory for Avaki Studio

described S:12
 setting S:10

WS API

accessibleDBOp API:42
 accessibleDS API:36
 accessiblePath API:19
 authentication API:9
 authorization API:9
 chmod API:19
 chown API:20
 data access API:2

WS API (continued)

data services SOAP operations API:34
 executeAdHocDBOp API:43
 executeAdHocDBOpWithOutput API:44
 executeAdHocDBOpWithOutputAttach API:46
 executeAdHocDBOpWithOutputString API:47
 executeDBOp API:48
 executeDBOpBytesInput API:49
 executeDBOpGridFileInput API:50
 executeDBOpWithOutput API:50
 executeDBOpWithOutputAttach API:52
 executeDBOpWithOutputString API:53
 executeDS API:36
 fileRead API:21
 fileReadAttach API:21
 fileReadString API:22
 fileWrite API:23
 getAttributes API:23
 getDBOpOutput API:54
 getDBOpOutputAttach API:55
 getDBOpParameters API:56
 getDBOpSchema API:56
 getDBOpSchemaAttach API:57
 getDBOpSchemaString API:58
 getDSOutput API:37
 getDSOutputAttach API:38
 getDSOutputString API:38
 getDSParameters API:39
 getOutputString API:55
 getSQL API:58
 getSystemAttributes API:24
 getUserAttributes API:24
 grid server API:6
 HTTP API:7, API:8
 HTTPS API:7, API:8
 isDSAvakiXML API:40
 listDBConns API:59
 listDBOps API:59
 listDBOpsByDBConn API:60
 listDomains API:25
 listDSs API:40
 listSearches API:25
 ls API:26
 lsSize API:26
 mkdir API:27
 mkdirParents API:27
 mkdirParentsServer API:28
 mkdirServer API:29
 mv API:29
 permissions API:30
 ports API:6
 removeAttribute API:31
 rm API:31
 search API:32
 setAttribute API:32
 tester API:33
 whoami API:33

WS clients, See web services clients

WSDLs
 about API:2
 as SOAP contracts API:3
 AvakiAPI.disco discovery file for .NET clients API:3
 AvakiAPIIDocLit.wsdl API:3
 AvakiAPIRpcEnc.wsdl API:3
 AvakiAPIWithMIMEDocLit.wsdl API:3
 AvakiAPIWithMIMERpcEnc.wsdl API:3
 choosing API:5
 document/literal API:3
 editing API:6
 locations of API:6
 provided by Avaki API:11
 rpc/encoded API:3
 standards compliance API:1
WSDLs directory O:35

X

X Window System libraries required for Avaki install on
 Unix AD:16
XA drivers, configuring for database connectors C:69, P:7, S:36
Xalan C:265, C:268, C:271, P:220, P:224, P:227
XAWorkHandler class for data services P:189
XAWorkUnit interface for data services P:189
XML data in Avaki O:11, P:273
XML indent size property AD:142
XML schema
 Avaki rowset
 class-name element P:279
 column-display-size element P:279
 column-index element P:279

XML schema (*continued*)
 core schema P:277
 overview P:277
 rowset-specific schema P:279
 sample schema P:280
data service
 class element P:261
 coherenceWindow element P:261
 dataService element P:262
 description element P:263
 initParameter element P:263
 inputParameter element P:264
 inputSource element P:265
 inputStream element P:266
 isList element P:266
 jarurl element P:267
 logicBox element P:268
 name element P:269
 outputStream element P:269
 ref element P:270
 target element P:270
 type element P:270
 urlLogicBox element P:271
 value element P:272
 values element P:272

XSLT
 in view generators, when not to use P:242
 using in data service plug-ins P:180
 See also Xalan, Saxon