

# **ART Administration Manual**

**3.0-rc1**

# Table of Contents

1. Introduction . . . . .	2
1.1. Administration Overview . . . . .	2
2. ART Database . . . . .	3
3. Settings . . . . .	4
4. Datasources . . . . .	7
4.1. Creating a new datasource . . . . .	7
4.2. Some JDBC Drivers and URLs . . . . .	8
5. Users . . . . .	11
5.1. Setting up the Public User . . . . .	11
5.2. Access Levels . . . . .	11
6. Authentication . . . . .	13
6.1. Internal Authentication . . . . .	13
6.2. Windows Domain Authentication . . . . .	13
6.3. LDAP Authentication . . . . .	13
6.4. Database Authentication . . . . .	13
6.5. CAS Authentication . . . . .	14
6.6. Custom Authentication Sources . . . . .	14
7. User Groups . . . . .	15
8. User Group Membership . . . . .	16
9. Report Groups . . . . .	17
10. Reports . . . . .	18
10.1. Creating a new report . . . . .	18
10.2. Report Types . . . . .	20
10.2.1. Tabular . . . . .	21
10.2.2. Tabular (html only) . . . . .	21
10.2.3. Crosstab . . . . .	21
10.2.4. Crosstab (html only) . . . . .	22
10.2.5. Tabular: Heatmap . . . . .	22
10.2.6. Charts . . . . .	22
10.2.7. XY . . . . .	23
10.2.8. Pie . . . . .	23
10.2.9. Bars/Stacked Bars/Line . . . . .	23
10.2.10. Time/Date Series . . . . .	23
10.2.11. Speedometer . . . . .	23
10.2.12. Bubble . . . . .	24
10.2.13. Heat Map . . . . .	24
10.2.14. Chart Options . . . . .	24
10.2.15. Group . . . . .	26
10.2.16. Update Statement . . . . .	26
10.2.17. Text . . . . .	26
10.2.18. Dashboard . . . . .	26
10.2.19. Dashboard: Gridstack . . . . .	27
10.2.20. JasperReports: Template Query . . . . .	27
10.2.21. JasperReports: ART Query . . . . .	27
10.2.22. JPivot: Mondrian . . . . .	27
10.2.23. JPivot: Mondrian XMLA . . . . .	28
10.2.24. JPivot: Microsoft XMLA . . . . .	28
10.2.25. jXLS: Template Query . . . . .	28

10.2.26.	jXLS: ART Query	28
10.2.27.	FreeMarker	28
10.2.28.	Thymeleaf	28
10.2.29.	ReactPivot	28
10.2.30.	PivotTable.js	29
10.2.31.	PivotTable.js: CSV Local	29
10.2.32.	PivotTable.js: CSV Server	29
10.2.33.	XDocReport: FreeMarker engine - Docx	29
10.2.34.	XDocReport: Velocity engine - Docx	29
10.2.35.	XDocReport: FreeMarker engine - ODT	30
10.2.36.	XDocReport: Velocity engine - ODT	30
10.2.37.	XDocReport: FreeMarker engine - PPTX	30
10.2.38.	XDocReport: Velocity engine - PPTX	30
10.2.39.	LOV: Dynamic	30
10.2.40.	LOV: Static	30
10.2.41.	Dynamic Job Recipients	30
10.2.42.	Dygraphs	31
10.2.43.	Dygraphs: CSV Local	31
10.2.44.	Dygraphs: CSV Server	31
10.2.45.	DataTables	31
10.2.46.	DataTables: CSV Local	32
10.2.47.	DataTables: CSV Server	32
10.2.48.	Fixed Width	32
10.2.49.	C3.js	34
10.2.50.	Chart.js	34
10.2.51.	Datamaps	35
10.2.52.	Datamaps: File	35
10.2.53.	Leaflet	36
10.2.54.	OpenLayers	36
10.2.55.	Saiku: Connection	36
10.2.56.	Saiku: Report	37
10.2.57.	MongoDB	37
10.3.	Parameters	40
10.3.1.	Single-Value Parameters	41
10.3.2.	Multi-Value Parameters	43
10.3.3.	Parameter Options	44
10.4.	Running a report via URL	44
10.4.1.	Report Options	44
10.4.2.	Chart Options	45
10.4.3.	Examples	46
10.5.	Report Formats	48
11.	Multiple Statements	50
12.	LOV Reports	51
12.1.	Example	51
13.	Dynamic SQL	53
13.1.	Using Groovy	53
13.1.1.	Example	53
13.2.	Using XML tags	53
13.2.1.	Operators	54
13.2.2.	Examples	55

14. Drill Down Reports . . . . .	57
14.1. Defining the main report . . . . .	57
14.2. Defining the drill down report . . . . .	57
14.3. Example . . . . .	57
14.4. Using the main report's parameters . . . . .	58
15. Dynamic Datasources . . . . .	59
16. RSS Feeds . . . . .	60
17. Dashboards . . . . .	62
17.1. Regular Dashboards . . . . .	63
17.1.1. Features . . . . .	63
17.1.2. Example Syntax . . . . .	64
17.2. Gridstack Dashboards . . . . .	65
17.2.1. Dashboard Properties . . . . .	65
17.2.2. Item Properties . . . . .	66
17.2.3. Example Syntax . . . . .	66
17.3. Tabbed Dashboards . . . . .	67
17.3.1. Example Syntax . . . . .	67
17.3.2. Tag Descriptions . . . . .	68
18. jXLS Reports . . . . .	69
19. JPivot Reports . . . . .	70
19.1. Mondrian . . . . .	70
19.1.1. Create the schema xml file . . . . .	70
19.1.2. Create a datasource . . . . .	70
19.1.3. Create the report . . . . .	70
19.1.4. Roles . . . . .	70
19.2. Mondrian XMLA . . . . .	70
19.2.1. Create the schema xml file . . . . .	71
19.2.2. Create the datasources.xml file . . . . .	71
19.2.3. Create a datasource . . . . .	71
19.2.4. Create the report . . . . .	71
19.2.5. Potential errors . . . . .	71
19.2.6. Accessing the Mondrian XMLA instance from MS Excel . . . . .	72
19.3. SQL Server XMLA . . . . .	72
19.3.1. Configure SSAS for HTTP access . . . . .	73
19.3.2. Provide access to the cube . . . . .	73
19.3.3. Create a datasource . . . . .	74
19.3.4. Create the report . . . . .	74
20. Rules . . . . .	75
21. Chained Parameters . . . . .	76
21.1. Example . . . . .	76
22. Scheduling . . . . .	78
22.1. Scheduling a new job . . . . .	78
22.2. Job Types . . . . .	78
22.3. Archives . . . . .	79
22.4. Job Auditing . . . . .	79
22.5. Saved Schedules . . . . .	80
22.6. Job Duration . . . . .	80
22.7. Shared Jobs . . . . .	80
22.7.1. Split Jobs . . . . .	80
22.8. Random Start Times . . . . .	80

23. <a href="#">Cached Results</a>	81
23.1. <a href="#">Create a new Cached Result</a>	81
23.2. <a href="#">Accessing a Cached Result</a>	81
24. <a href="#">Dynamic Recipients</a>	83
24.1. <a href="#">Dynamic Recipients only</a>	83
24.2. <a href="#">Dynamic Recipients + Personalization</a>	83
24.3. <a href="#">Dynamic Recipients + Filtering</a>	84
25. <a href="#">FTP Servers</a>	85
26. <a href="#">Integrated Windows Authentication</a>	86
26.1. <a href="#">Prerequisites</a>	86
26.2. <a href="#">On the Active Directory server</a>	86
26.3. <a href="#">On the Application server</a>	86
26.4. <a href="#">On the client machine</a>	88
26.4.1. <a href="#">Omitting the credentials box</a>	88
26.5. <a href="#">Using a keytab file</a>	89
26.6. <a href="#">Changing the spnego user</a>	90
27. <a href="#">CAS Authentication</a>	91
28. <a href="#">Application Logs</a>	92
28.1. <a href="#">SQL Logging</a>	92
29. <a href="#">Tomcat Configuration</a>	93
29.1. <a href="#">Memory options</a>	93
29.1.1. <a href="#">Windows</a>	93
29.1.2. <a href="#">Linux</a>	93
29.2. <a href="#">Accessing Tomcat via Apache</a>	93
29.2.1. <a href="#">Using mod_proxy</a>	94
29.2.2. <a href="#">Using mod_proxy_ajp</a>	94
29.2.3. <a href="#">Using mod_jk</a>	94
30. <a href="#">Customizing ART</a>	96
30.1. <a href="#">Using live files</a>	96
30.1.1. <a href="#">Setting up Apache Ant</a>	96
30.1.2. <a href="#">Customizing java files</a>	96
30.1.3. <a href="#">Customizing jsp files</a>	96
30.1.4. <a href="#">Customizing other files</a>	96
30.1.5. <a href="#">Updating jasperreports</a>	97
30.2. <a href="#">Using Ant</a>	97
30.2.1. <a href="#">Using Ant without an IDE</a>	97
30.2.2. <a href="#">Using Ant with NetBeans</a>	97
30.3. <a href="#">Using Maven</a>	98
30.3.1. <a href="#">Using Maven without an IDE</a>	98
30.3.2. <a href="#">Using Maven with NetBeans</a>	99
30.3.3. <a href="#">Using Maven with Eclipse</a>	99
30.4. <a href="#">Custom Export Path</a>	100
30.5. <a href="#">Translating ART</a>	100
31. <a href="#">Support</a>	102

## **ART Administration Manual**

# 1. Introduction

ART is a Java web application that enables quick deployment of SQL query results.

- **ART Administrators** define datasources, reports, etc.
- **ART Users** run reports and view the results in a browser or in a variety of file formats.

ART is open source software distributed under the [GPLv3 license](#). You can install and use it without any charge.

## 1.1. Administration Overview

The Administrators define a number of items including

- **ART Database** used to store ART objects e.g. report definitions
- **Settings** used to configure how ART works
- **Datasources** against which reports are run
- **Report Groups** used to group reports
- **Reports** that can be run
- **User Groups** used to group users
- **Users** who can access the application
- **User Group Membership** to add or remove users from user groups
- **Access Rights** to determine which users or user groups can access which reports, report groups or jobs
- **Admin Rights** to determine which report groups and datasources administrators can manage
- **Parameters** to be used by reports
- **Rules** to be applied to reports
- **Rule Values** to be used with specific users or user groups
- **Jobs** to run reports at scheduled times
- **Schedules** to manage schedules that can be used when creating jobs
- **FTP Servers** to enable ftping of job output
- **Caches** to enable clearing of ART object caches
- **Connections** to view datasource connection pools
- **Loggers** to manage which application debug information is logged by ART

The typical process when defining a report is

- Obtain the SQL statement for the report
- Create the report
- Grant access rights to allow users to run the report

## 2. ART Database

The ART database is the database used by ART to hold details like users, report definitions etc. Use the **Configure | ART Database** menu to define connection details for the ART database.

Attribute	Description
<b>Database Type</b>	The database software that the database runs on
<b>JNDI</b>	Defines whether the ART database is a JNDI datasource
<b>JDBC Driver</b>	JDBC driver name
<b>JDBC URL</b>	JDBC URL. If you are using a JNDI datasource, set this to the JNDI name of your datasource e.g. <code>jdbc/MyDatasource</code> . You can also use the full JNDI url e.g. <code>java:comp/env/jdbc/MyDatasource</code>
<b>Username</b>	Database username. The user needs SELECT, INSERT, UPDATE and DELETE rights on the ART tables.
<b>Password</b>	Database password
<b>Test SQL</b>	Short SQL query used to determine if a connection is alive e.g. "Select 1"
<b>Connection Pool Timeout</b>	How long an idle connection should be maintained in the connection pool before being closed. This setting applies to the ART database as well as all report datasources.
<b>Max Pool Connections</b>	The maximum number of connections a connection pool can open to the same datasource. Further requests are queued. This setting applies to the ART database as well as all report datasources.
<b>Connection Pool Library</b>	The connection pool library to use. This setting applies to the ART database as well as all report datasources.



### 3. Settings

Certain settings are used to configure how ART works. Use the **Configure | Settings** menu to manage ART settings.

Setting	Description
<b>SMTP Server</b>	Host name for the email server used to send emails
<b>SMTP Port</b>	Port on which SMTP server is listening
<b>Use StartTLS</b>	Defines whether to use the StartTLS protocol when sending emails
<b>Use SMTP Authentication</b>	Defines whether the SMTP server requires a username and password in order to send emails
<b>SMTP User-name</b>	Username to be used when sending emails if the email server is configured to require SMTP authentication
<b>SMTP Password</b>	Password to be used when sending emails if the email server is configured to require SMTP authentication
<b>Default Authentication Method</b>	The authentication method that will be used by ART
<b>Windows Domain Controller</b>	Used with windows domain authentication. The IP address of the windows domain controller.
<b>Allowed Windows Domains</b>	Used with windows domain authentication. The domain name of the windows domain used for authentication. Multiple domains can be specified, each separated by a comma.
<b>Database Authentication JDBC Driver</b>	Used with database authentication. The JDBC driver for the database used for authentication.
<b>Database Authentication JDBC URL</b>	Used with database authentication. The JDBC URL for the database used for authentication.
<b>LDAP Server</b>	Used with LDAP authentication. IP address of the LDAP server.
<b>LDAP Port</b>	Used with LDAP authentication. LDAP server port.
<b>LDAP Connection Encryption Method</b>	Used with LDAP authentication. Defines which protocol to use for the LDAP connection.
<b>LDAP URL</b>	Used with LDAP authentication. LDAP server URL. If the LDAP server and port fields have been used, leave this setting blank. This setting only provides an alternative way of specifying the location of the LDAP server.

Setting	Description
<b>Use Anonymous Bind</b>	Used with LDAP authentication. Defines whether to use anonymous bind when connecting to the LDAP server in order to search for and authenticate users.
<b>LDAP Bind DN</b>	Used with LDAP authentication. The DN to use when connecting to the LDAP server in order to search for and authenticate users.
<b>LDAP Bind Password</b>	Used with LDAP authentication. The password to use when connecting to the LDAP server in order to search for and authenticate users.
<b>LDAP User ID Attribute</b>	Used with LDAP authentication. The LDAP attribute which will be used to match ART usernames.
<b>LDAP Authentication Method</b>	Used with LDAP authentication. The authentication method to be used with the LDAP server
<b>LDAP Realm</b>	Used with LDAP authentication. The LDAP realm when using the Digest-MD5 authentication method. If blank, the default realm will be used.
<b>Default Max Rows</b>	The default maximum number of rows to output for a report
<b>Specific Max Rows</b>	The maximum number of rows to output for specific report formats, defined as a comma separated list of settings with each setting in the format view-mode:value e.g. htmlGrid:5000,xls:10000. Report formats are case sensitive.
<b>PDF Font Name</b>	Name of a custom font that should be used in generation of pdf output, and charts. For jasper reports, custom fonts need to be defined in the jrxml file. See the <a href="#">Tips</a> documentation for details on how to use custom fonts with jasper reports.
<b>PDF Font File</b>	Path to a font file that contains the custom font
<b>PDF Font Directory</b>	Path to a directory that contains font files, one of which may be used in the pdf font name field
<b>PDF Font Encoding</b>	Encoding to use for the custom font
<b>PDF Font Embedded</b>	Whether the custom font should be embedded in the generated pdf output
<b>Administrator Email</b>	Email address which is displayed in link at the bottom of ART web pages
<b>Date Format</b>	Format to be used for date portions of dates. Format strings to be used is as per the java SimpleDateFormat class. See <a href="http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html">http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html</a>
<b>Time Format</b>	Format to be used for time portion of dates. Format strings to be used is as per the java SimpleDateFormat class. See <a href="http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html">http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html</a>

<b>Setting</b>	<b>Description</b>
<b>Report Formats</b>	The report formats that will be available to users when they run a report, defined as a comma separated list. Report format names are case sensitive and the order specified will be respected in the list shown to users.
<b>Display Null</b>	Defines whether the string "null" is displayed where a column in a report is null
<b>Max Running Reports</b>	The maximum number of reports that can be running at any one time
<b>Show Header in Public User Session</b>	Defines whether to show the menu bar and page footer for reports that are run with public access
<b>Mondrian Cache Expiry Period</b>	Number of hours after which the mondrian cache is automatically cleared. Set to 0 to disable automatic clearing.
<b>Scheduling Enabled</b>	Defines whether scheduled jobs will run.
<b>RSS Link</b>	RSS URL if ART will be used to generate RSS feeds
<b>Max File Upload Size (MB)</b>	Maximum file upload size for jasperreports, jXLS or mondrian files. Set to -1 for no limit.
<b>ART Base URL</b>	The base URL for ART e.g. <a href="http://art-server:8080/art">http://art-server:8080/art</a> . This is used to include a link to publish job output in publish job reminder emails.

## 4. Datasources

A datasource is a database against which you want to run reports. Use the **Configure | Datasources** menu to manage datasources.

Datasources

+ Add

Edit

Delete

Show 10 entries

Search:

	ID	Name	Description	Active	Action
	<input type="text" value="ID"/>	<input type="text" value="Name"/>	<input type="text" value="Description"/>	<input type="text" value="Active"/>	
<input type="checkbox"/>	1	SampleDB		Active	<div><div> Edit</div><div> Delete</div></div>
<input type="checkbox"/>	2	ArtRepository		Active	<div><div> Edit</div><div> Delete</div></div>

Showing 1 to 2 of 2 entries

First

Previous

1

Next

Last

- Use the **Add** button to create a new datasource
- Use the **Edit** and **Delete** buttons at the top to edit or delete multiple datasources and the buttons on the side to edit or delete individual datasources
- Use the **Search** field to search for a particular datasource. You can also search by individual columns using the appropriate search fields below the column headings.
- You can sort the datasources list by clicking on the column headings

### 4.1. Creating a new datasource

From the Datasources page, use the Add button and then specify the settings for the datasource. ART can run reports against any datasource for which a JDBC driver is available.

Attribute	Description
<b>Name</b>	A name to identify the datasource
<b>Description</b>	A description for the datasource
<b>Active</b>	Defines whether the datasource is available for use
<b>Datasource Type</b>	Defines whether this is a JDBC based datasource or an OLAP datasource
<b>JNDI</b>	Defines whether the datasource is a JNDI datasource
<b>Database Type</b>	The database software that the database runs on
<b>JDBC Driver</b>	JDBC driver name
<b>JDBC URL</b>	JDBC URL of the target database. If you are using a JNDI datasource, set this to the JNDI name of your datasource e.g. <code>jdbc/MyDatasource</code> , or the full JNDI url e.g. <code>java:comp/env/jdbc/MyDatasource</code>
<b>Username</b>	Database user on the target database. This user should have minimum rights on the database and tables you plan to use in your reports, mostly only <b>SELECT</b> rights to the relevant tables.
<b>Password</b>	Password for the database user
<b>Test SQL</b>	Short SQL query used to determine if a connection is valid e.g. "Select 1"
<b>Connection Pool Timeout</b>	How long a connection should be maintained in the connection pool before being closed

**Note:**

- The Database Type field is not saved. It is only there as a guide to help fill the JDBC Driver and URL fields.
- Connections do not always close gracefully - for example if a network outage occurs a broken connection might stay in the pool and would throw an error when used. The **Test SQL** query is used every **Timeout** minutes to validate the connection. If it does not execute successfully, the connection is closed and removed from the pool.

## 4.2. Some JDBC Drivers and URLs

Database: **CUBRID**

Driver Name: `cubrid.jdbc.driver.CUBRIDDriver`

JDBC URL: `jdbc:cubrid:<server_name>:<port>:<database_name>[:?<property1>=<value1>[&<property2>=<value2>][&...]]` (default port is 33000)

Driver Available from: <http://www.cubrid.org>

Database: **Oracle**

Driver Name: oracle.jdbc.OracleDriver

JDBC URL: jdbc:oracle:thin:@<server\_name>:<port>:<sid> (default port is 1521)

Driver Available from: <http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html>

Database: **MySQL**

Driver Name: com.mysql.jdbc.Driver

JDBC URL: jdbc:mysql://<server\_name>[:port]/<database\_name>[?<property>=<value>[&...]] (default port is 3306)

Driver Available from: <http://dev.mysql.com/downloads/connector/j/>

Database: **MariaDB**

Driver Name: org.mariadb.jdbc.Driver

JDBC URL: jdbc:mariadb://<server\_name>[:port]/<database\_name>[?<property>=<value>[&...]] (default port is 3306)

Driver Available from: Driver Available from:  
<https://mariadb.com/kb/en/mariadb/about-mariadb-connector-j/>

Database: **PostgreSQL**

Driver Name: org.postgresql.Driver

JDBC URL: jdbc:postgresql://<server\_name>[:port]/<database\_name>[?<property>=<value>[&...]] (default port is 5432)

Driver Available from: <http://jdbc.postgresql.org/>

Database: **SQL Server (Microsoft driver)**

Driver Name: com.microsoft.sqlserver.jdbc.SQLServerDriver

JDBC URL: jdbc:sqlserver://<server\_name>[:port];database-Name=<database\_name>[;instanceName=<instance\_name>][;<property>=<value>[;...]] (default port is 1433)

Driver Available from: <http://msdn.microsoft.com/en-us/data/aa937724.aspx>

Database: **SQL Server (JTDS driver)**

Driver Name: net.sourceforge.jtds.jdbc.Driver

JDBC URL:  
jdbc:jtds:sqlserver://<server\_name>[:port]/<database\_name>[;instance=<instance\_name>][;<property>=<value>[;...]] (default port is 1433)

Driver Available from: <http://jtds.sourceforge.net>

Database: **HSQldb (Standalone mode)**

Driver Name: org.hsqldb.jdbcDriver

JDBC URL: jdbc:hsqldb:file:<file\_path>[;shutdown=true;hsqldb.write\_delay=false;create=false][;<property>=<value>[;...]

Driver Available from: <http://hsqldb.org/>

Database: **HSQldb (Server mode)**

Driver Name: org.hsqldb.jdbcDriver

JDBC URL:  
jdbc:hsqldb:hsqldb://<server\_name>[:port]/<database\_alias>[;<property>=<value>[;...]  
(default port is 9001)

Driver Available from: <http://hsqldb.org/>

Database: **DB2 (IBM Data Server Driver for JDBC and SQLJ)**

Driver Name: com.ibm.db2.jcc.DB2Driver

JDBC URL: jdbc:db2://<server\_name>[:port]/<database\_name>[:<property>=<value>[;...]] (default port for IBM DB2 is many times one of the following: 446, 6789, or 50000)

Driver Available from: <http://www-01.ibm.com/support/docview.wss?rs=4020&uid=swg21385217>  
(registration required).

This driver can connect to a DB2 database on any platform i.e. iSeries, zSeries, or Intel-based machines with Linux or Windows.

Database: **DB2 for iSeries (Toolbox driver)**

Driver Name: com.ibm.as400.access.AS400JDBCdriver

JDBC URL: jdbc:as400://<server\_name>;prompt=false;translate  
binary=true[;<property>=<value>[;...]]

Driver Available from: <http://jt400.sourceforge.net/>

**Note:**

- ART comes with drivers for CUBRID, PostgreSQL, MySQL, MariaDB, HSQldb, SQL Server (jTDS driver), SQL Server (Microsoft driver) so you don't need to download these drivers

## 5. Users

Use the **Configure | Users** menu to manage users.

ID	Username	Full Name	Active	Action
1	admin	The Mighty Admin	Active	Edit Delete
2	auser	A Poor User (but can schedule jobs!)	Active	Edit Delete

- Use the **Add** button to create a new user
- Use the **Edit** and **Delete** buttons at the top to edit or delete multiple users and the buttons on the side to edit or delete individual users
- Use the **Search** field to search for a particular user. You can also search by individual columns using the appropriate search fields below the column headings.
- You can sort the users list by clicking on the column headings

### Note:

- It is always possible to log into ART by specifying the ART database username and password

### 5.1. Setting up the Public User

If you want a report to be run by anyone, without authentication, you can define a user with the special username **public\_user** and grant this user access to the report. This is useful for availing reports without requiring users to log in to ART

### 5.2. Access Levels

Users are assigned access levels depending on the permissions they require within the application. Each higher access level has the permissions of all access levels below it.



<b>Level</b>	<b>Permissions</b>
<b>Normal User</b>	Run reports, view jobs, view archives
<b>Schedule User</b>	Schedule jobs
<b>Junior Admin</b>	Create reports, view application documentation
<b>Mid Admin</b>	Manage access rights
<b>Standard Admin</b>	Create users, create user groups, define user group membership, manage admin user rights, manage jobs
<b>Senior Admin</b>	Create datasources, create report groups, create rules and rule values, create schedules, manage caches, connections and loggers, view application logs
<b>Super Admin</b>	Manage ART database and application settings

## 6. Authentication

Use the **Configure | Settings** menu to define the authentication method to be used by ART.

### 6.1. Internal Authentication

Internal authentication authenticates users with a username and password combination defined and stored within ART. Users can change their passwords using the **Password** menu.

### 6.2. Windows Domain Authentication

If the users are in a windows domain environment, you can have them log in to ART using the usernames and passwords they use to log in to windows.

Do the following to configure ART to use windows domain authentication.

- In the Settings page, set the **Default Authentication Method** to **Windows Domain**
- Set the **Windows Domain Controller** field to the domain controller machine name (IP address should also work). You can do the following to get the domain controller machine name.

```
ping <domain name> (This will get the ip address of the domain controller)
ping -a <ip address of domain controller> (This will get the hostname of the domain controller)
```

- Set the **Allowed Windows Domains** field to the windows domain name. Use all uppercase letters for the domain name

For each user who requires access to ART, you will also need to create a user within ART with the same username as their windows login username. Grant them appropriate access to reports. The users can now log in to ART using their windows username and password.

### 6.3. LDAP Authentication

Do the following to configure ART to use LDAP authentication.

- In the Settings page, set the **Default Authentication Method** to **LDAP**
- Provide values for the LDAP fields. If you fill the ldap server and port fields, leave the url field blank. If you use the url field, leave the server and port fields blank.

For each user who requires access to ART, you will also need to create a user within ART with the same username as their uid (or samAccountName for Active Directory). Grant them appropriate access to reports. The users can now log in to ART using their ldap uid and password.

### 6.4. Database Authentication

Database logins can also be used to access ART. Do the following to use the database authentication that comes with ART.

- Have or create database users on any database. The users need to be able to connect to the database but don't need to have any other rights on the database
- In the Settings page, set the **Default Authentication Method** to **Database**

- Fill the Database Authentication JDBC fields. The JDBC driver for the database will need to be available in the **ART\_HOME\WEB-INF\lib** directory or **TOMCAT\_HOME\lib** directory if using Apache Tomcat.

For each user who requires access to ART, you will also need to create a user within ART with the same username as their database login username. Grant them appropriate access to reports. The users can now log in to ART using their database username and password.

## 6.5. CAS Authentication

ART can use an existing CAS server to do the authentication of users. To use CAS authentication, take the following steps

- Ensure the CAS server is running
- Start ART with Internal authentication configured
- Ensure the CAS users are also created within ART. The password within ART is not important as CAS will be doing the authentication.
- In the Settings page, change the **Default Authentication Method** to **CAS**
- Enter the **CAS Logout URL** e.g. `https://cas-server:8443/cas/logout` to enable users to log out of CAS, and thereby log out of all applications that use CAS.
- Log out of ART
- Stop ART
- Edit ART's **web.xml** file and uncomment the items in the CAS configuration section, providing appropriate urls for the CAS and ART applications.
- Start ART again
- You should be redirected to the CAS login page where you will provide appropriate credentials, after which you will be redirected back to ART

For each user who requires access to ART, you will also need to create a user within ART with the same username as their CAS login username. Grant them appropriate access to reports.

## 6.6. Custom Authentication Sources

Additional, custom authentication sources can be used by creating a jsp page to check the user credentials - for example by executing a query on a remote database - and then redirecting to the reports page.

This is the code fragment to redirect an authenticated user to ART.

```
if ( <CONDITION> ) {
    // Authentication Successful!
    session.setAttribute("username", username);
    response.sendRedirect("/art/reports");
}
```

The user needs to be defined in ART. ART will check if the username exists, and if this is the case, will let the user proceed.

## 7. User Groups

Use the **Configure | User Groups** menu to manage user groups.

User groups are used to logically treat a set of users as one entity. If a group is granted access to a report, members of that group get access to the report. This allows for easier management of access rights. A user can belong to zero, one or many user groups.

## 8. User Group Membership

Use the **Configure | User Group Membership** menu to manage user group membership.

## 9. Report Groups

Use the **Configure | Report Groups** menu to manage report groups.

Report groups are used to logically associate reports.

**Note:**

- In a standard organization, you may want to define several groups to match the different business areas of the company e.g. INVENTORY, PURCHASING, FINANCE etc.
- In order to create Development, QA and Production environments, you can create several groups e.g. INVENTORY\_PROD, INVENTORY\_QA, and INVENTORY\_DEV, and have a newly developed report created in the INVENTORY\_DEV group, moved to the INVENTORY\_QA group for user tests and finally to INVENTORY\_PROD once testing is complete.
- It is possible to limit the report groups that an administrator can deal with. You can therefore have administrators who deal only with a development environment and others who deal with the production environment.

# 10. Reports

Use the **Configure | Reports** menu to manage reports.

## 10.1. Creating a new report

Use the Add button from the Reports Configuration page to create a new report.

Field	Description
<b>ID</b>	Auto-generated ID used to identify the report
<b>Name</b>	Name of the report
<b>Report Group</b>	Report Group to which the report will be belong
<b>Active</b>	Whether the report can be run or not
<b>Hidden</b>	Whether the report is listed to users or not
<b>Short Description</b>	Short description for the report. For charts, this will appear as the title of the chart.
<b>Description</b>	Longer description for the report
<b>Contact Person</b>	Can be used to store the name of the contact or reference person for the report
<b>Type</b>	The type of report
<b>Datasource</b>	The datasource against which the report will be executed
<b>Uses Rules</b>	This specifies if the report will use rules
<b>Parameters In Output</b>	This determines if the selected parameter values should be displayed in the report output
<b>Display Resultset</b>	The resultset to display if the sql source contains multiple sql statements. Leave as 0 if the sql source doesn't have multiple statements. Set to 1 to use the first statement, 2 to use the second, etc. Set to -1 to use the <b>select</b> statement, regardless of how many statements exist. Set to -2 to use the <b>last</b> statement, regardless of how many statements exist. Your RDBMS may not support multiple statements in a query or may require some configuration for it to work. See the <b>Multiple Statements</b> section for more details.
<b>Default Report Format</b>	The default report format that should be selected for this report. Please note that not all report formats are available for all report types.
<b>Hidden Columns</b>	For tabular reports, a comma separated list of column indices or column names for columns that should not be included in the generated output

Field	Description
<b>Total Columns</b>	For tabular reports, a comma separated list of column indices or column names for columns that should be totalled in the generated output
<b>Date Format</b>	For tabular reports, the format that should be used for date columns. Leave blank to use the default. The format is as per <a href="#">Java SimpleDateFormat format</a> .
<b>Number Format</b>	For tabular reports, the format that should be used for numeric columns. Leave blank to use the default.
<b>Column Formats</b>	For tabular reports, the formats that should be used for specific date or numeric columns. Each column specification comes in a new line. The specification is in the format {column index or column name}:{format}. Examples 1:dd/MM/yyyy or due_date:dd MMM yyyy or amount:#,##0.00. For xls and xlsx report formats, one cannot use individual column formats so the formats specified in the Date Format and Number Format fields would be used for all respective columns.
<b>Locale</b>	For tabular reports, the locale to be used when formatting dates and numbers e.g. de or en_US. Leave blank to use the default. For xls and xlsx report formats, this setting has no effect.
<b>Null Number Display</b>	For tabular reports, a string that should be used for numeric columns where the value is null. Leave blank to use the default. For xls and xlsx report formats, this setting has no effect.
<b>Null String Display</b>	For tabular reports, a string that should be used for string columns where the value is null. Leave blank to use the default.
<b>Fetch Size</b>	The number of rows retrieved by the database driver per trip to the database. Leave as 0 to use the driver's default. This value represents a trade-off between memory and query execution time and may be useful to set for reports that return large amounts of data in order to avoid out-of-memory errors. Not all databases honour this setting.
<b>Page Orientation</b>	The orientation to be used in generated documents e.g. with pdf output
<b>x-axis Label</b>	For charts, the x axis label
<b>y-axis Label</b>	For charts, the y axis label
<b>Width</b>	For charts, the chart width
<b>Height</b>	For charts, the chart height
<b>Background Colour</b>	For charts, the background colour
<b>y-axis Min</b>	For charts, the minimum value for the y-axis. Leave as 0 to use the full data range.
<b>y-axis Max</b>	For charts, the maximum value for the y-axis. Leave as 0 to use the full data range.
<b>Rotate x-axis labels at</b>	For charts, the number of categories at which the labels will be displayed vertically instead of horizontally. Set to 1 to always display labels vertically.



Field	Description
<b>Remove x-axis labels at</b>	For charts, the number of categories at which labels are omitted from the chart. Set to 1 to always omit labels.
<b>Secondary Charts</b>	For charts, a comma separated list of report IDs of other charts which are to be displayed with this one. These charts will use separate y-axes. Some chart types cannot have secondary charts, including pie charts, speedometer, bubble and heat map charts.
<b>Template</b>	For jasper reports, select the .jrxml file to use. For pivot table (mondrian) reports, select the mondrian cube .xml file to use. For jXLS reports, select the .xls or .xlsx template file to use. For FreeMarker reports, select the .ftl template file to use. For XDocReport reports, select the .docx, .odt or .pptx template file to use. For React-Pivot reports, specify the .js file that contains the react pivot configuration for the report.
<b>XMLA Data-source</b>	For mondrian via xmla, the xmla datasource name as it appears in the <b>data-sources.xml</b> file on the xmla server
<b>XMLA Catalog</b>	For mondrian or mondrian via xmla, the catalog name as it appears in the mondrian cube xml file. For SSAS, the database name.
<b>Options</b>	Provides additional options depending on the report type. The options are specified in JSON format with different report types expecting different JSON specifications.
<b>Source</b>	The SQL query used to retrieve the required data. It can contain parameter placeholders, specified using labels (#parameter_name#). XML-style elements can be used to create Dynamic SQL statements. Some special tags can be used in the query (:USERNAME:, :DATE:, :TIME:)

**Note:**

- Three special tags can be used within the SQL statement and are substituted at runtime with their respective values:

```
:USERNAME: - replaced by the username of the user who is executing the query
:DATE: - replaced by the current date (format YYYY-MM-DD)
:TIME: - replaced by the current time (format YYYY-MM-DD HH:MI:SS)
```

- You can use a stored procedure to return query results. When defining the query, in the SQL source section, use the syntax for calling a stored procedure in the RDBMS you are using e.g. For MySQL you can use something like "call my\_sp". Another RDBMS may use syntax like "exec my\_sp".

## 10.2. Report Types

### 10.2.1. Tabular

A tabular result exportable to spreadsheet, pdf etc.

### 10.2.2. Tabular (html only)

A tabular result that can only be displayed in HTML format. This may be used to embed HTML code in the SQL query in order to modify display colours of certain columns etc. To do this, concatenate the SQL with the required HTML tags. e.g. For MySQL, to display positive values in green and negative values in red, you can use something like

```
SELECT col1,
CASE WHEN int_col2>0 THEN
concat('<div style="background-color: green">',cast(int_col2 as char),'</div>')
ELSE
concat('<div style="background-color: red">',cast(int_col2 as char),'</div>')
END as "My Formatted Column",
col3
from my_table
```

### 10.2.3. Crosstab

Rearranges the query output into crosstab format, exportable to spreadsheet, pdf etc. If you want values to be summed, include the summing in the SQL query e.g. select year,quarter,sum(amount) from orders group by year,quarter

The SQL result set is expected to have either 3 or 5 columns:

```
SELECT xAxisCol "xAxisLabel", yAxisCol "yAxisLabel", Value FROM ...
(data type: string, string, any)
```

OR

```
SELECT xAxisCol, xAxisAltSort, yAxisCol, yAxisAltSort, Value FROM ...
(data type: string, string, string, string, any)
```

The AltSort columns are used to sort the x-axis (rows) and y-axis (columns). The following helps to illustrate this.

<pre>/* input */ // A Jan 14 // A Feb 24 // A Mar 34 // B Jan 14 // B Feb 24 // C Jan 04 // C Mar 44 //</pre>	<pre>/* input */ A 1 Jan 1 14 A 1 Feb 2 24 A 1 Mar 3 34 B 2 Jan 1 14 B 2 Feb 2 24 C 3 Jan 1 04 C 3 Mar 3 44 ^-----^-----Used to sort the x/y axis</pre>
<pre>/* output */ //      y-axis //        // x-axis - _   Feb Jan Mar</pre>	<pre>/* output */ //      y-axis //        // x-axis - _   Jan Feb Mar</pre>

```
//      A      24   14   34      A      14   24   34
//      B      24   14   -      B      14   24   -
//      C      -    04   44      C      04   -    44
//      ^--- Jan comes after Feb!
```

Without the AltSort columns, as in the first output, Feb appears before Jan. This is because the string "Feb" is alphabetically before "Jan". However, Jan should appear before Feb because that is how they appear in the order of months. You can therefore use the month number in the sort column to ensure that Jan is displayed before Feb. Another example would be if you are displaying dates in a format like "Apr-2011" (MMM-YYYY). Apr-2011 is alphabetically before Jan-2011, so the alternative sort column could be set to YYYY-MM format (e.g. 2011-04) to order the period names in the right way.

### 10.2.4. Crosstab (html only)

Same as crosstab but limited to HTML output only

### 10.2.5. Tabular: Heatmap

Displays numeric information using shades of colour depending on the values. A number of options can be specified in the **Options** field to configure the output.

Property	Data Type	Description
<b>columns</b>	optional - array of integers	The columns that should be coloured. If not specified, all columns are included in the colouring. If any cell in the applicable columns contains a non-numeric value, no colouring will be done. An error will be displayed in the javascript console for the web browser indicating that <b>hex is undefined</b> .
<b>colors</b>	optional - array of strings	Colours that should be applied, beginning with colours for the lower values going up. The colour strings are specified in hex format e.g. #ffffff. For example for a two colour scheme in a red range, one can specify the setting as "colors": [ "#ffffff", "#ff4a53" ]
<b>perColumn</b>	optional - boolean (default = false)	If true, each applicable column is graded and coloured separately. If false, the whole table is considered when grading the colours.

### 10.2.6. Charts

Display the results as a chart, exportable to pdf or png.

The layout of the SQL must follow a specific syntax for each different type of chart

### 10.2.7. XY

```
SELECT Value1, Value2 "SeriesName" FROM ...  
(data type: number, number )
```

Dynamic Series

```
SELECT Value1, Value2, SeriesName FROM ...  
(data type: number, number, string)
```

### 10.2.8. Pie

```
SELECT Category, Value FROM ...  
(data type: string, number )
```

### 10.2.9. Bars/Stacked Bars/Line

Static Series

```
SELECT Item, Value1 "SeriesName1" [, Value2, ...] FROM ...  
(data type: string, number [, number, ...] )
```

Dynamic Series

```
SELECT Item, SeriesName, Value FROM ...  
(data type: string, string, number)
```

Example:

```
SELECT Product, Region, SUM(VOLUME) FROM sales group by product,region
```

### 10.2.10. Time/Date Series

Static Series

```
SELECT Timestamp|Date, Value1 "SeriesName1" [, Value2, ...] FROM ...  
(data type: timestamp|date, number, [, number, ...] ). Timestamp/Dates must be unique.
```

Dynamic Series

```
SELECT Timestamp|Date, SeriesName, Value FROM ...  
(data type: timestamp|date, string, number). Timestamp/Dates must be unique.
```

Example:

```
SELECT ORDER_DATE, PRODUCT, SUM(VOLUME) FROM orders group by order_date,product
```

### 10.2.11. Speedometer

```
SELECT DataValue, MinValue, MaxValue, UnitsDescription [, Range1, Range2, ...] FROM ...  
(data type: number, number, number, string)  
Ranges represent optional columns and each range has 3 values separated by : i.e.  
RangeUpperValue:RangeColour:RangeDescription (data type: number, string, string).  
RangeUpperValue can be a percentage.
```

Example:

```
SELECT reading, 0, 100, "degrees",
```

```
"50:#00FF00:Normal",
"80%:#FFFF00:Warning",
"100:#FF0000:Critical"
FROM temperature_reading
```

## 10.2.12. Bubble

```
SELECT Value1, Value2 "SeriesName", Value3 [, normalisedValue3] FROM ...
(data type: number, number, number [, number] )
```

## 10.2.13. Heat Map

```
SELECT Value1, Value2, Value3 [, Option1, Option2, ...] FROM ...
(data type: number, number, number [, string, string, ...] )
```

Example:

```
SELECT x, y, z, "upperBound=100"
FROM myvalues
```

## 10.2.14. Chart Options

Certain options can be specified in the **Options** field to configure the chart output.

Property	Data Type	Description
<b>seriesColors</b>	optional - object	Defines specific colours to be used for given chart series. Each property of the object consists of the series ID as the property name and the hex colour code as the property value. Multiple properties can be specified e.g. "seriesColors": { "0" : "#ff4a53", "2" : "#4a538a" }. The first series has an ID of 0.
<b>dateFormat</b>	optional - string	For date series and time series charts, defines the date format to be used on the x-axis, overriding the one automatically assigned by the chart.

### Note:

- ART uses the cewolf and jfreechart libraries to generate charts. These libraries in turn use standard java AWT to plot charts. In order to work correctly, AWT needs a graphic display. If you are using a "headless" workstation (i.e. a Unix box without X) you need to start the JVM with the option **-Djava.awt.headless=true**
- You can specify specific colours to be used for the series displayed in a chart by adding columns to the query resultset in the format `seriesColor:<series index>:<hex color code>` e.g. for a pie chart,

```
SELECT description, volume, "seriesColor:0:#ff8000"
FROM orders
```

- For **bubble charts**, the size of the bubbles is determined by Value3. The actual size of the bubbles as they appear in the chart is also relative to the values on the y axis (Value2). If your values for Value3 are much larger than those for Value2 for example, you may find the whole chart filled in one colour, with no bubbles. In this case, you can increase the range of the y axis by setting the **y-axis Min** and **y-axis Max** fields for the query. Alternatively, you can include an

extra column in the result set that contains a normalised value that will be used to scale the size of the bubbles e.g.

```
SELECT Value1, Value2 "MySeries", Value3,
Value3 * (SELECT max(Value2) FROM mytable)/(SELECT max(Value3) FROM mytable)
FROM mytable
```

- For **heat map** charts, the following options are available. Option names and values are case sensitive. Options are defined in the result set with a column value whose syntax is <option>=<value> i.e. the option name and value separated by =

Option	Possible Values	Default	Description
<b>upperBound</b>	Any number, positive or negative	1	Highest value displayed. If you don't specify this option, the chart may appear blank if all your values are above 1.
<b>lowerBound</b>	Any number, positive or negative	0	Lowest value displayed
<b>scalePos</b>	top left bottom right	bottom	Default position of the colour scale relative to the chart
<b>scaleLabel</b>	Any string	None	Title the colour scale
<b>scaleTextPos</b>	topleft topright bottomleft bottomright	topleft	Position of the scale title relative to the colour scale
<b>scaleBorder</b>	Hex colour code e.g. #00E000	None	Colour of border displayed around the colour scale box
<b>stripWidth</b>	Any positive integer	10	Width/thickness of the colour scale
<b>subdivisions</b>	Any positive integer	10	For grey scale or a 2 colour scheme, i.e. if no color#n options are configured, the number of shades of grey or shades of the 2 colour scheme to use
<b>color#&lt;n&gt;</b> e.g. color#1, color#2	<number> : <hex colour code> e.g. 0.0 : #FF0000	None	The colour to use for a given range of values. The colour is determined by Value3 of the result set. The number is the lower bound of the range. The option value has the lower bound and colour separated by : e.g. to display 0-10 in red and 11-50 in green, you'll define two columns in the result set, "color#1=0:#FF0000", "color#2=11:#00FF00"

Option	Possible Values	Default	Description
<b>lowerColor</b>	Hex colour code	None	The colour of the lowest value if you want a 2 colour scheme with a linear gradient from the lower colour to the upper colour e.g. set lowerColor to white (#FFFFFF) and upperColor to red (#FF0000) if you want values to be represented as shades of red
<b>upperColor</b>	Hex colour code	None	The colour of the highest value if you want a 2 colour scheme with a linear gradient from the lower colour to the upper colour e.g. set lowerColor to white (#FFFFFF) and upperColor to red (#FF0000) if you want values to be represented as shades of red

### 10.2.15. Group

Groups the report data.

For example, if a query's tabular output looks like:

AA	AA	B	C	D
AA	AA	E	F	G
AA	BB	H	J	K
AA	BB	X	Y	Z

using "Group on 2 columns", the data would be grouped by the first 2 columns, and the output would look like:

AA	AA
B	C   D
E	F   G
AA	BB
H	J   K
X	Y   Z

For a "Group on n columns" report, the query must be ordered by the the first n-1 columns.

### 10.2.16. Update Statement

Used to execute statements that do not return any rows e.g. INSERT/UPDATE/DELETE statements

### 10.2.17. Text

Used to define a piece of text that can be displayed

### 10.2.18. Dashboard

Used to display reports in a single portal-like page

### 10.2.19. Dashboard: Gridstack

Used to display reports in a single portal-like page, allowing for specifying exact positioning of report items within the dashboard, specifying the height and width that the report items should occupy, allowing for resizing the items dynamically using the mouse, and moving the items around using drag-and-drop.

### 10.2.20. JasperReports: Template Query

To generate formatted reports e.g. documents given to customers, you can use JasperReports. To create a jasper report, use the [JasperSoft Studio](#) report designer.

Displays a jasper report based on the selected jrxml template. The query within the jasper report template will be used to generate the results. The query will be run against the selected datasource. You can define parameters that will be passed to the jasper report. Jasper report parameters are defined with a specific type. The following mapping of ART and jasperreport parameter types should be used.

ART parameter type	JasperReports parameter type
<b>Varchar, Text</b>	java.lang.String
<b>Date, DateTime</b>	java.util.Date
<b>Integer</b>	java.lang.Integer
<b>Double</b>	java.lang.Double
<b>Multi-Value parameters</b>	java.util.List

If the report contains a subreport, set the **Subreport Expression** property of the subreport object to the file name of the subreport with a .jasper extension e.g. "**subreport1.jasper**" (include the quotes). When creating the report in ART, upload the main report's .jrxml file using the main template field and upload the subreport's .jrxml file using the resources field. If there are multiple subreports, you can select multiple files and upload all of them at once using the **Start upload** button. You can also select files by dragging from the file explorer and dropping to the browser page.

### 10.2.21. JasperReports: ART Query

Displays a jasper report based on the selected jrxml template. The query as defined in the SQL source will be used to generate the results. The query will be run against the selected datasource.

### 10.2.22. JPivot: Mondrian

Used to provide OLAP (slice and dice) analysis using the mondrian engine. The MDX for the query should be put in the source section and the xml file that defines the mondrian cube should be selected in the template field. More details about JPivot reports can be found in the JPivot Reports section of the manual.



### **10.2.23. JPivot: Mondrian XMLA**

Used to provide OLAP analysis by accessing a mondrian server via the xmla protocol.

### **10.2.24. JPivot: Microsoft XMLA**

Used to provide OLAP analysis by accessing an SQL Server Analysis Services server via the xmla protocol.

### **10.2.25. jXLS: Template Query**

Displays a report in MS Excel format(xls or xlsx) based on a pre-formatted jXLS template. The query within the jXLS template will be used to generate the results. The query will be run against the selected datasource. More details about jXLS reports can be found in the jXLS Reports section of the manual.

### **10.2.26. jXLS: ART Query**

Displays a report in MS Excel format(xls or xlsx) based on a pre-formatted jXLS template. The query as defined in the SQL source will be used to generate the results. The query will be run against the selected datasource.

### **10.2.27. FreeMarker**

Displays a report in the browser based on a freemarker template - <http://freemarker.org/>. The query as defined in the SQL source will be used to generate the results. The data can be accessed in the template by iterating over the **results** variable. Report parameters are also available using the parameter name e.g. **\${param1.value}**. An example report is included in the demo database.

### **10.2.28. Thymeleaf**

Displays a report in the browser based on a thymeleaf template - <http://www.thymeleaf.org/>. The query as defined in the SQL source will be used to generate the results. The data can be accessed in the template by iterating over the **results** variable. Report parameters are also available using the parameter name e.g. **\${param1.value}**. An example report is included in the demo database.

### **10.2.29. ReactPivot**

Generates a pivot-table-like report based on tabular data using the ReactPivot library - <https://davidguttman.github.io/react-pivot/>. You will need to create a javascript file that contains the react pivot configuration for the report. You need to define **dimensions**, **reduce** and **calculations** options. Other options can also be defined to override the defaults. The demo database also comes with some samples of this report type. This report type is useful when you want summary and analysis of numeric data by category.

### 10.2.30. PivotTable.js

Generates a pivot table based on data from a database - making use of the PivotTable.js library - <https://github.com/nicolaskruchten/pivottable>. A javascript file containing custom configuration can be uploaded using the **Template** field. The demo database contains a sample report of this type. Performance should be acceptable up to 100,000 rows, depending on the user's computer.

### 10.2.31. PivotTable.js: CSV Local

Enables generation of a pivot table using data from a user supplied csv file. Additionally, a javascript file can be specified in the **Template** field with custom configuration options. Any delimited file can be used, not only csv.

### 10.2.32. PivotTable.js: CSV Server

Generates a pivot table based on data from a specific csv file uploaded to the server. This file can be uploaded using the **Add files** button. The file name will also need to be specified in the **Options** field with a format similar to the following. A **dataFile** attribute is expected, whose value is a string - the name of the csv file.

```
{
    "dataFile": "file_name.csv"
}
```

Additionally, a javascript template file can be specified in the **Template** field with custom configuration options. Any delimited file can be used, not only csv.

### 10.2.33. XDocReport: FreeMarker engine - Docx

You can use MS Word (docx), LibreOffice Writer (odt) or MS PowerPoint (pptx) documents as a template for reports. You simply create the document as you would like it to appear and then insert fields that will be replaced with data at runtime. This uses the [XDocReport](https://github.com/opensagres/xdocreport/wiki) library. You can have a look at the XDocReport wiki, <https://github.com/opensagres/xdocreport/wiki>, for examples on how to create templates for different needs. The query results will be contained in a variable named **results** which can be used in the FreeMarker or Velocity expressions.

Displays a report based on the selected docx template, which uses freemarker syntax - <http://freemarker.org/docs/index.html> for expressions within the document. The query as defined in the SQL source will be used to generate the results.

### 10.2.34. XDocReport: Velocity engine - Docx

Displays a report based on the selected docx template, which uses velocity syntax - <https://velocity.apache.org/engine/devel/user-guide.html> for expressions within the document. The query as defined in the SQL source will be used to generate the results.

### **10.2.35. XDocReport: FreeMarker engine - ODT**

Displays a report based on the selected odt template, which uses freemarker syntax. The query as defined in the SQL source will be used to generate the results.

### **10.2.36. XDocReport: Velocity engine - ODT**

Displays a report based on the selected odt template, which uses velocity syntax. The query as defined in the SQL source will be used to generate the results.

### **10.2.37. XDocReport: FreeMarker engine - PPTX**

Displays a report based on the selected pptx template, which uses freemarker syntax. The query as defined in the SQL source will be used to generate the results.

### **10.2.38. XDocReport: Velocity engine - PPTX**

Displays a report based on the selected pptx template, which uses velocity syntax. The query as defined in the SQL source will be used to generate the results.

### **10.2.39. LOV: Dynamic**

Creates an LOV query whose values are based on an sql query. The query can have either one column, or two columns if the parameter value and the text displayed to the user needs to be different

```
SELECT city_id, city_name
FROM cities
```

OR

```
SELECT city_id
FROM cities
```

### **10.2.40. LOV: Static**

Creates an LOV query whose values are set in the sql source section. Values are separated by new lines. If the value and display text need to be different, separate the two with a |

```
Toaster
pr-01|Laptops
pr-02|Desktops
```

### **10.2.41. Dynamic Job Recipients**

Defines a query that can be used to specify dynamic recipients when scheduling a job. It can have either one column, or multiple columns. In either case, the first column must contain email addresses for the recipients.

## 10.2.42. Dygraphs

Generates a time series chart using data from a database - making use of the dygraphs library - <http://dygraphs.com/>. A javascript file containing custom configuration can be uploaded using the **Template** field. The demo database contains some sample reports of this type. The query must have a minimum of two columns. The first can be a date or number field, and will be the data on the x-axis. The second column must be a number and will represent the first series data. Additional columns of number type can be included, which will be data for additional series. There should be no record with a null date in the query result. If you zoom into the data by clicking and dragging, you double-click on the chart to zoom out.

## 10.2.43. Dygraphs: CSV Local

Enables generation of a dygraphs chart using data from a user supplied csv file. Additionally, a javascript file can be specified in the **Template** field with custom configuration options. Any delimited file can be used, not only csv. The first column of the data, if a date, should have dates formatted as `YYYY/MM/dd` e.g. 2017/02/17, and datetime data formatted as `YYYY/MM/dd HH:mm` e.g. 2017/02/17 16:02 or `YYYY/MM/dd HH:mm:ss` e.g. 2017/02/17 16:02:59. The csv data should not contain any quotes.

## 10.2.44. Dygraphs: CSV Server

Generates a dygraphs chart based on data from a specific csv file uploaded to the server. This file can be uploaded using the **Add files** button. The file name will also need to be specified in the **Options** field with a format similar to the following. A **dataFile** attribute is expected, whose value is a string - the name of the csv file.

```
{
  "dataFile": "file_name.csv"
}
```

Additionally, a javascript template file can be specified in the **Template** field with custom configuration options. Any delimited file can be used, not only csv.

## 10.2.45. DataTables

Enables display of data from a database in a table using the DataTables library - <https://datatables.net/>. A javascript file containing custom configuration can be uploaded using the **Template** field. The demo database contains some sample reports of this type. In the **Options** field, there are a number of attributes that can be specified.

Property	Data Type	Description
<b>outputDateFormat</b>	optional - string (default = blank)	The format of output data for date columns. If blank, the data will be output as is. The format is as per <a href="#">Java SimpleDateFormat format</a> .
<b>outputDateTimeFormat</b>	optional - string (default = blank)	The format of output data for datetime columns. If blank, the data will be output as is. The format is as per <a href="#">Java SimpleDateFormat format</a> .
<b>showColumnFilters</b>	optional - boolean (default = true)	Determines if column filters will be displayed

### 10.2.46. DataTables: CSV Local

Enables display of data from a user supplied csv file in a DataTables table. Additionally, a javascript file can be specified in the **Template** field with custom configuration options. Any delimited file can be used, not only csv.

### 10.2.47. DataTables: CSV Server

Enables display of data from a specific csv file uploaded to the server. This file can be uploaded using the **Add files** button. The file name will also need to be specified in the **Options** field with a format similar to the following. A **dataFile** attribute is expected, whose value is a string - the name of the csv file.

```
{
  "dataFile": "file_name.csv"
}
```

Additionally, a javascript template file can be specified in the **Template** field with custom configuration options. Any delimited file can be used, not only csv.

### 10.2.48. Fixed Width

Generates fixed width output, primarily for jobs that require output in fixed width format. The sql query determines which data will be in the output. The **Options** field determines the formatting of this output and has the following possible properties.

Property	Data Type	Description
<b>fieldLengths</b>	array of integers	Used to define the field lengths of the output e.g. [ 5 , 6 ] means that the first field will have a length of 5 and the second field will have a length of 6

Property	Data Type	Description
<b>fieldLengthsByName</b>	array of objects	Each object specifies the field name and the field length e.g. [ { "order_id": 8 }, { "order_date": 14 } ] means the order_id field will have a length of 8 and the order_date field will have a length of 14. You must specify either the <b>fieldLengths</b> or the <b>fieldLengthsByName</b> property.
<b>includeHeaders</b>	optional - boolean (default = true)	Determines whether the column names will be included in the output as the first row
<b>defaultAlignmentForHeaders</b>	optional - string	Either <b>left</b> , <b>right</b> or <b>center</b> . If present, overrides any field alignment set and uses the given alignment for the header columns
<b>padding</b>	optional - string (default = space)	A single character defining the default padding to use for fields. Must not be the empty string.
<b>useDefaultPaddingForHeaders</b>	optional - boolean (default = true)	Determines if the headers should use the global <b>padding</b> setting or use the padding set for individual fields
<b>dateFormat</b>	optional - string	Determines the format that should be used for DATE columns
<b>dateTimeFormat</b>	optional - string	Determines the format that should be used for DATETIME columns
<b>numberFormat</b>	optional - string	Determines the format that should be used for numeric columns
<b>fieldDateFormats</b>	optional - array of objects	Each object specifies the date format and an array of fields that should use that format e.g. [ { "dd/MM/yyyy": [ "order_date", "due_date" ] }, { "dd-MMM-yyyy": [ "payment_date" ] } ]
<b>fieldNumberFormats</b>	optional - array of objects	Each object specifies the number format and an array of fields that should use that format e.g. [ { "0.00": [ "amount" ] }, { "#,##0.##": [ "volume" ] } ]

Property	Data Type	Description
<b>fieldAlignmentByName</b>	optional - array of objects	Each object specifies the alignment to be used and an array of field names that should use that alignment e.g. [ { "right": [ "amount", "volume" ] } ]. The alignment is either <b>left</b> , <b>right</b> or <b>center</b> .
<b>fieldAlignmentByPosition</b>	optional - array of objects	Each object specifies the alignment to be used and an array of field positions that should use that alignment e.g. [ { "right": [ 2, 5 ] } ]. The alignment is either <b>left</b> , <b>right</b> or <b>center</b> . The first field in the output is marked as position 0.
<b>fieldPaddingByName</b>	optional - array of objects	Each object specifies the padding to be used and an array of field names that should use that padding e.g. [ { "_" : [ "amount", "volume" ] } ]. The padding should be a single character.
<b>fieldPaddingByPosition</b>	optional - array of objects	Each object specifies the padding to be used and an array of field positions that should use that padding e.g. [ { "_" : [ 2, 5 ] } ]. The padding should be a single character. The first field in the output is marked as position 0.

The demo database has a few sample fixed width reports and jobs.

### 10.2.49. C3.js

Generates charts using the c3.js library - <http://c3js.org/>. A javascript file containing chart configuration needs to be uploaded using the **Template** field. The demo database contains some sample reports of this type. If you would like to use custom css, provide a json object in the **Options** field, setting the **cssFile** property to the file name of the css file to use. This file can be uploaded using the **Add files** option.

```
{
  "cssFile": "file_name.css"
}
```

### 10.2.50. Chart.js

Generates charts using the chart.js library - <http://www.chartjs.org/>. A javascript file containing chart configuration needs to be uploaded using the **Template** field. The demo database contains some sample reports of this type. In the **Options** field, you can specify the **width** (integer - default = 300) or **height** (integer - default = 100) properties to define a custom width or height for the chart.

## 10.2.51. Datamaps

Enables display of data on a map, using the Datamaps library - <https://datamaps.github.io/>, using data from a database. A javascript file containing the datamaps configuration needs to be uploaded using the **Template** field. In the **Options** field, there are a number of attributes that can be specified.

Property	Data Type	Description
<b>datamapsJsFile</b>	string	The file name of the datamaps javascript file. Appropriate files can be downloaded from <a href="https://github.com/markoh/datamaps/tree/master/dist">https://github.com/markoh/datamaps/tree/master/dist</a> .
<b>width</b>	optional - string (default = 500px)	CSS width for the div that contains the map
<b>height</b>	optional - string (default = 300px)	CSS height for the div that contains the map
<b>dataFile</b>	optional - string	The file name of the json or csv file that contains data elements for the map. This file can be uploaded using the <b>Add files</b> option.
<b>dataType</b>	optional - string (default = json)	Either <b>csv</b> if the dataFile specified is in csv format or <b>json</b> if it's in json format.
<b>mapFile</b>	optional - string	The file name of a file that contains the custom map in TopoJSON format. When using this option, use the <b>datamaps.none.min.js</b> datamaps js file. This file can be uploaded using the <b>Add files</b> option. It is important to note that if using the Add files option to upload files, files with multiple dots in the name are not allowed, so such files need to be renamed e.g. datamaps-none-min.js
<b>cssFile</b>	optional - string	The file name of a css file that contains custom css to be used with the display. This file can be uploaded using the <b>Add files</b> option.

If using a custom map and you have a shapefile, you can **Import** the zip file with shapefile specification files to <http://mapshaper.org/> and then use the **Export** option to convert the shapefile to **TopoJSON**. You will then use the TopoJSON file for the report.

## 10.2.52. Datamaps: File

Enables display of data on a map, using the Datamaps library, using data from a csv or json file. You can also use this report type if the data is hardcoded within the template file. Requirements and options are similar to those of the **Datamaps** report type. The demo database contains some sample reports of this type.



### 10.2.53. Leaflet

Enables display of data on a web map or tiled map using the Leaflet library - <http://leafletjs.com/>. A javascript file containing the leaflet configuration needs to be uploaded using the **Template** field. In the **Options** field, there are a number of attributes that can be specified.

Property	Data Type	Description
<b>height</b>	optional - string (default = 400px)	CSS height for the div that contains the map
<b>cssFile</b>	optional - string	The file name of a css file that contains custom css to be used with the display. This file can be uploaded using the <b>Add files</b> option.
<b>dataFile</b>	optional - string	The file name of a file that contains extra data to be used with the map e.g. a GeoJSON file. This file can be uploaded using the <b>Add files</b> option.
<b>jsFiles</b>	optional - array of strings	File names of javascript files that need to be incorporated for the map display e.g. leaflet plugins. The files can be uploaded using the <b>Add files</b> option.
<b>cssFiles</b>	optional - array of strings	File names of additional css files to be used with the display. The files can be uploaded using the <b>Add files</b> option.

### 10.2.54. OpenLayers

Enables display of data on a web map using the OpenLayers library - <http://openlayers.org/>. A javascript file containing the openlayers configuration needs to be uploaded using the **Template** field. In the **Options** field, there are a number of attributes that can be specified. These attributes are the same as those for Leaflet reports.

### 10.2.55. Saiku: Connection

Provides a connection that will be used to generate cubes in the Analytics view. ART uses the [Saiku](#) library to provide the Analytics functionality.

You would first create a **Datasource** from the **Configure | Datasources** menu. The datasource will need to be of either the Olap4J Mondrian or Olap4j XMLA type. You would then fill the other data-source details as appropriate and then specify this datasource in the **Datasource** field when creating the saiku connection report. After doing this, if you click on the **Analytics** menu and then the **Add query** icon, you will see in the **Cubes** drop down, your connection and the cubes it has.

If you use an Olap4j Mondrian datasource, you can specify different roles that different users will have when using the connection. To do this, create a rule for the saiku connection report. Use a dummy column name. Next, specify rule values for that rule for different users. If you need to specify multiple roles, add multiple rule values.

Also for Olap4j Mondrian datasources, you can omit the **Catalog** part from the JDBC URL and instead upload the catalog (schema xml file) to use in the **Template** field. Note that when you test or save an olap4j mondrian datasource without specifying the Catalog, you will get an error. You can

ignore this error if you will be specifying the Catalog in the saiku connection.

It is also worth noting that saiku uses mondrian 4, and as such if using olap4j mondrian, or mondrian xmla datasources, it is best if the schema xml files are in mondrian 4 structure.

### 10.2.56. Saiku: Report

When you perform some analysis from the Analytics view, you can save that analysis. When you do this, this is saved as a Saiku: Report type. The report source for this report type is a JSON representation of a saiku QueryModel. You can have a look at it to see what is generated. It isn't recommended to build this by hand.

Once you have created a saiku report, you can give users access to it like with other report types. The users will also need to be given access to the saiku connection that is the basis for the saiku report. Users will see a list of saiku reports they have access to by clicking on the **Open query** icon at the top of the Analytics view. Users can then run the reports, edit and save new reports, and delete reports they have created. Users cannot overwrite or delete reports for which they don't have exclusive access i.e. they are the only ones who have access to the report. Only junior admins and above can overwrite or delete any saiku report.

### 10.2.57. MongoDB

Allows you to query a MongoDB database. To use it, first create a datasource of type **MongoDB**. Set the URL as appropriate. Next, create the report, setting the report **Type** as **MongoDB** and selecting the mongodb datasource in the **Datasource** field. In the report source field, enter the groovy script that will be used to retrieve data from the mongodb database. The syntax of querying can either be that of the [MongoDB Java Driver](#), the [Jongo](#) library or the [Morphia](#) library. ART passes a MongoClient connection object to the groovy script with the variable name **mongoClient**. This is the variable that is to be used as the connection to the mongodb server. ART also passes any report parameters specified using the parameter name as the variable name.

If it is required to display the returned results in a table, the groovy script must return a List of objects that are to be displayed. Also, the groovy script is run through a sandbox and only classes contained in the **WEB-INF\groovy-whitelist.txt** file will be allowed to be used within the script. If you use a class that is not contained in the whitelist, a **SecurityException** error will occur when you run the report. If you get such an error, add the class mentioned to the whitelist and run the report again.

#### Example using MongoDB Java Driver syntax

```
import org.bson.Document;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;

MongoDatabase db = mongoClient.getDatabase("test");
MongoCollection<Document> collection = db.getCollection("mycol");
MongoCursor<Document> cursor = collection.find().iterator();

List<Document> records = new ArrayList<>();
try {
    while (cursor.hasNext()) {
        records.add(cursor.next());
    }
}
```

```

    }
} finally {
    cursor.close();
}

```

```
return records;
```

### Example using Jongo syntax

```

import com.mongodb.DB;
import org.bson.Document;

import org.jongo.Jongo;
import org.jongo.MongoCollection;
import org.jongo.MongoCursor;

DB db = mongoClient.getDB("test");

Jongo jongo = new Jongo(db);
MongoCollection friends = jongo.getCollection("mycol");
MongoCursor<Object> cursor = friends.find("{title: 'MongoDB'}").as(Object.class);

List<Object> records = new ArrayList<>();
try {
    while (cursor.hasNext()) {
        records.add(cursor.next());
    }
} finally {
    cursor.close();
}

return records;

```

### Example using Morphia syntax

```

package art.groovy

import org.mongodb.morphia.annotations.Entity;
import org.mongodb.morphia.annotations.Id;
import org.mongodb.morphia.Datastore;
import org.mongodb.morphia.Morphia;
import org.mongodb.morphia.query.Query;
import org.bson.types.ObjectId;

@Entity(value = "employeesCol", noClassnameStored = true)
class Employee{
    @Id
    ObjectId id
    String name
    String department
}

Morphia morphia = new Morphia();

morphia.map(Employee.class);

Datastore datastore = morphia.createDatastore(mongoClient, "employeesDb");

```

```
Query<Employee> query = datastore.createQuery(Employee.class);
List<Employee> records = query.asList();

return records;
```

For **Morphia** queries, you need to specify a package name for the groovy script. This package needs to be **art.groovy** in order not to get a security exception. The other alternative to not specifying this package name is to turn off the groovy sandbox within the **WEB-INF\art-custom-settings.json** file. Changes to this file require the application to be restarted in order to take effect.

In the **Options** field, there are a number of attributes that can be specified.

Property	Data Type	Description
<b>columns</b>	optional - array of strings	Names of columns that should be included in the tabular output. This may be useful when querying documents that may have different attributes. If not specified, ART will use the first document retrieved to determine the columns that will be included in the output.
<b>columnDataTypes</b>	optional - array of objects	Each object specifies a column name and the data type that is to be used for that column e.g. [ { "col1": "numeric" } ]. The possible data types are <b>string</b> , <b>numeric</b> , <b>date</b> and <b>datetime</b> . The default is <b>string</b> so if a column is to be treated as a general string, it doesn't need to be specified.
<b>inputDateFormat</b>	optional - string (default = yyyy-MM-dd)	The format of input data for columns with a <b>date</b> specification. The format is as per <a href="#">Java SimpleDateFormat format</a> .
<b>outputDateFormat</b>	optional - string (default = blank)	The format of output/displayed data for columns with a <b>date</b> specification. If it's blank, the data will be output/displayed as is and no formatting will be done. The format is as per <a href="#">Java SimpleDateFormat format</a> .
<b>inputDateTimeFormat</b>	optional - string (default = yyyy-MM-dd HH:mm:ss.SSS)	The format of input data for columns with a <b>datetime</b> specification. The format is as per <a href="#">Java SimpleDateFormat format</a> .
<b>outputDateTimeFormat</b>	optional - string (default = blank)	The format of output/displayed data for columns with a <b>datetime</b> specification. If it's blank, the data will be output/displayed as is and no formatting will be done. The format is as per <a href="#">Java SimpleDateFormat format</a> .
<b>showColumnFilters</b>	optional - boolean (default = true)	Determines if column filters will be displayed in the tabular output.

Another alternative for querying MongoDB is to install and use [Apache Drill](#). Drill allows you to use normal SQL to query a MongoDB database. When using drill, you wouldn't use the MongoDB data-source type or the MongoDB report type. You would use the JDBC datasource type with the drill driver, and then use any of the other report types e.g. Tabular etc. The drill jdbc driver can be obtained from the drill installation and copied to the ART\_HOME\WEB-INF\lib directory.

**Note:**

- If the output for a report creates a file e.g. pdf, this file is stored in the **WEB-INF\work\export** directory

## 10.3. Parameters

Parameters enable different output to be generated depending on values selected or input by a user.

Field	Description
<b>ID</b>	Auto-generated ID used to identify the parameter
<b>Name</b>	Name of the parameter as it appears in reports SQL source
<b>Description</b>	Description for the parameter
<b>Parameter Type</b>	Whether the parameter will have a single value ( <b>Single-Value</b> ) or whether it can have multiple values ( <b>Multi-Value</b> )
<b>Label</b>	The label for the parameter in the select parameters page
<b>Help Text</b>	Help text displayed as a tooltip in the select parameters page
<b>Placeholder Text</b>	Placeholder text displayed in the field in the select parameters page
<b>Data Type</b>	The type of data that the parameter can hold
<b>Date Format</b>	For <b>Date</b> and <b>DateTime</b> data types, the date format to use. Leave blank to use the default. The format is as per <a href="#">Java SimpleDateFormat format</a> .
<b>Default Value</b>	The value for the parameter to use when the parameter is first displayed in the select parameters page, or the value to use if the report is run by direct url and the parameter is not specified within the url. For <b>Multi-Value</b> parameters, multiple values can be specified, with each value on a new line.
<b>Default Value Report</b>	A report that provides the default values. This needs to be an LOV report.
<b>Shared</b>	Whether the parameter is available for selecting when defining report parameters, through the <b>Add Existing</b> option.
<b>Hidden</b>	Whether the parameter is displayed in the select parameters page
<b>Use LOV</b>	Whether the parameter uses a dropdown to provide a list of values to choose from in the select parameters page
<b>LOV Report</b>	If <b>Use LOV</b> is specified, the report that generates the available values. This needs to be an LOV report.
<b>Drilldown Column Index</b>	The index of the column that will be set with the parameter's value when used with drilldown reports. The first column has an index of 1.
<b>Options</b>	Options used to further configure the parameter. The options are specified in JSON format.

### 10.3.1. Single-Value Parameters

Single-value parameters are used to pass single values to the SQL query. The parameter's name surrounded by the # character is used to identify the parameter within the SQL source. For example, the following query has two parameters.

```
SELECT *
FROM orders
WHERE order_date > #date#
AND vendor_name like #vendor# -- you do not need to put the ' character around the name
```

Take the following steps to use a single-value parameter

- Create the report, putting the parameter placeholder in the appropriate place within the SQL
- After saving the report, click on the **Parameters** button for the just saved report
- Click on the **Add New** button
- Fill in the appropriate fields for the parameter
- In the **Parameter Name** field, specify the parameter name as used in the SQL source e.g. date for the example above. Avoid having spaces in parameter names.
- **Save** the parameter
- You can also use the **Add Existing** button to use a parameter that was created previously and marked as **Shared**.

**Note:**

- When editing report parameters from the **Report Parameters** page, clicking on the link with the parameter name enables you to edit core parameter definition attributes, while clicking on the Edit button allows you to edit extra attributes specifically related to that report parameter.
- The same parameter (#parameter\_name#) can be used several times in the same query. All occurrences will be substituted with the same value.
- Avoid using the special labels **#rules#** or **#recipient#**. These labels are used for the rules and dynamic job recipients functionality respectively, and using them for your parameters may result in your query not working as expected.
- There is a slight difference between Varchar and Text parameters. Varchar parameters provide a text box for input of a few characters while Text parameters provide a text area for input of much longer strings.
- For Date and DateTime parameters, the following values can be used

Value	Meaning
<b>now</b>	The date and time when the report is run
<b>today</b>	The date when the report is run, with the time being 00:00
<b>add days, weeks, months, years, hours, minutes, seconds, milliseconds increment</b>	The date when the report is run plus the given increment e.g. add days 1 for the run date plus one day, add months -1 for the run date minus one month.

- **Workaround for boolean parameters**

There is no boolean data type for single-value parameters because different RDBMSs implement the boolean data type differently. As a workaround, you can create a static LOV report to return the boolean states true and false, as used in your database, and use that LOV report to provide the boolean parameter values.

Example main query

```
SELECT * FROM mytable WHERE my_boolean_column=#boolean_param#
```

Example static LOV query for boolean values

```
true|True  
false|False
```

Lastly, create a **Single-Value** parameter named `boolean_param`, with data type as **Varchar**, set **Use LOV to Yes** and set the **LOV Report** to the report created for this purpose above.

If the boolean states are "1" and "0" instead of "true" and "false", you can create a static LOV query like the one below and set the parameter data type to **Integer** instead of Varchar.

```
1|True  
0|False
```

### 10.3.2. Multi-Value Parameters

Multi-value parameters are used to pass multiple values to the SQL query. To define a multi-value parameter, put the WHERE...IN clause in the desired location in your query and include the parameter placeholder to indicate where the values will go e.g.

```
SELECT *  
FROM orders  
WHERE product_name IN(#product_list#)
```

Take the following steps to use a multi-value parameter

- Create the report, putting the WHERE...IN clause and parameter placeholder in the appropriate place within the SQL
- Create the parameter definition using the same process as with single-value parameters
- Now when a user runs the query, they will be presented with a parameters section where they can select one or more values for the parameter. An "All" option will be displayed to specify all values.

#### X-Parameter Definitions

There exists an alternative syntax for specifying multi-value parameters. This provides for defining IN and NOT IN clauses where IN clauses take the format `$x{in,<column_name>,<parameter_name>}` and NOT IN clauses take the format `$x{notin,<column_name>,<parameter_name>}` e.g.

```
SELECT *  
FROM orders  
WHERE $x{in,product_name,product_list}
```

It should be noted that with x-parameter definitions, the parameter name is case sensitive and must be specified exactly the way it is defined in the parameter configuration. Also, a different separator other than a comma can be used to separate the different components of the x-parameter definition e.g. ; or | etc.



X-parameter definitions would particularly be used when running reports against Oracle databases and the multi-value parameter is expected to have more than 1000 items. Using the normal syntax would result in an error like the following **ORA-01795: maximum number of expressions in a list is 1000**.

**Note:**

- If the values for a multi-value parameter are not derived from an LOV, the end user will enter the multiple values required in the text box provided, with each value put on a new line.
- When setting the **Default Value** for a multi-value parameter, several values can be set as defaults by entering each value on a separate line in the field provided. Default values can also come from an LOV report that can be specified in the **Default Value Report** field.

### 10.3.3. Parameter Options

When defining a parameter, a number of options can be specified in the **Options** field. These are defined in JSON format with the following possibilities.

Property	Data Type	Description
<b>i18n</b>	optional - object	Allows for localization of parameter configurations. It can have the properties <b>label</b> , <b>helpText</b> , <b>defaultValue</b> , <b>placeholderText</b> , with each property being an array of objects providing the locale and the appropriate localized string. e.g. <pre>"i18n": { "label": [{ "en, sw": "Test Label" }, { "pt": "Another Label" } ], "helpText": [{ "lt": "Sample help text" } ] }</pre>

## 10.4. Running a report via URL

It is possible to run a report directly via URL. The report format and parameters are included in the URL. The basic URL for running a report is something like  
`http://server_name:port/art/runReport?reportId=x&other_parameters`

If direct URL access is needed without requiring the user to log in to ART beforehand, the special user **public\_user** must be granted access to the query and **&public=true** must be included in the URL.

### 10.4.1. Report Options

Some report options can be specified in the URL to determine how the run should be handled.

Option	Data Type	Comments
<b>p-{parameter_name}</b>	String	Specifies report parameters e.g. <code>p-duedate=2010-05-06</code>
<b>reportFormat</b>	String	Specifies the report format that should be used for the output generated by the report e.g. <code>reportFormat=pdf</code> . Not all report formats are available for all report types.
<b>splitColumn</b>	Integer	For group reports, the group column i.e. the column index by which grouping should be done e.g. <code>splitColumn=2</code>
<b>public</b>	Boolean	If present, determines that the user doesn't need to be logged in to ART to view the report results e.g. <code>public=true</code> . If authentication is required, don't include this parameter, even with a value of false.
<b>showSelectedParameters</b>	Boolean	If present, indicates that the used or selected report parameters should be included in the report output e.g. <code>showSelectedParameters=true</code> . Omit the parameter entirely if this is not required.
<b>showSql</b>	Boolean	If present, indicates that the final SQL used to generate the report results is shown in the browser e.g. <code>showSql=true</code> . Omit the parameter entirely if this is not required.
<b>allowSelectParameters</b>	Boolean	If present, indicates that the parameters box should be displayed in the browser to allow the user to re-run the report by changing the reports parameter values.
<b>startSelectParametersHidden</b>	Boolean	If present, indicates that the parameters box should start as hidden/collapsed. The parameters box can then be displayed by clicking on the <b>Parameters</b> button.
<b>prettyPrint</b>	Boolean	For <b>json</b> and <b>jsonBrowser</b> report formats, determines if the json output should be pretty printed.

### 10.4.2. Chart Options

For chart reports, additional options specific to charts can be specified in the URL.

Option	Data Type	Comments
<b>showLegend</b>	Boolean	If present, indicates that the legend should be shown e.g. <code>showLegend=true</code> . Omit the parameter entirely if this is not required.
<b>showLabels</b>	Boolean	If present, indicates that data labels should be shown e.g. <code>showLabels=true</code> . Omit the parameter entirely if this is not required.
<b>showData</b>	Boolean	If present, indicates that the data used to generate the chart should be shown e.g. <code>showData=true</code> . Omit the parameter entirely if this is not required.
<b>showPoints</b>	Boolean	If present, indicates that data points on the chart should be highlighted e.g. <code>showPoints=true</code> . Omit the parameter entirely if this is not required.
<b>rotateAt</b>	Integer	Indicates the number of categories after which the x-axis labels will be displayed vertically instead of horizontally e.g. <code>rotateAt=1</code> . This may aid in the readability of the labels for some charts.
<b>removeAt</b>	Integer	Indicates the number of categories after which the x-axis labels will be removed completely e.g. <code>removeAt=10</code> .
<b>chartWidth</b>	Integer	Indicates the width of the chart in pixels e.g. <code>chartWidth=1000</code> . The maximum is 2048.
<b>chartHeight</b>	Integer	Indicates the height of the chart in pixels e.g. <code>chartHeight=700</code> . The maximum is 1024.
<b>yAxisMin</b>	Double	Indicates the minimum value of the y-axis e.g. <code>yAxisMin=-10.5</code> .
<b>yAxisMax</b>	Double	Indicates the maximum value of the y-axis e.g. <code>yAxisMax=100</code>
<b>backgroundColor</b>	String	Indicates the hex colour code for the background color of the chart e.g. <code>backgroundColor=#FFFFFF</code> .
<b>labelFormat</b>	String	Indicates the format of data labels e.g. <code>labelFormat={0} = {1} ({2})</code> or <code>labelFormat=off</code> . If set to "off", this indicates that labels should not be shown. For pie charts, <code>{0}</code> will display the pie section key, <code>{1}</code> will display the absolute section value and <code>{2}</code> will display the percent amount of the pie section. Setting it to <code>{0} = {1} ({2})</code> would display a string like "Laptops = 17 (42%)" for a section of the pie chart showing laptops numbering 17 and having a percentage value of 42%. For bar charts, use <code>{2}</code> to display the data value.

### 10.4.3. Examples

**Example 1:** (single-value parameters)

```
http://server_name:port/art/runReport?
reportId=120&reportFormat=html&p-startdate=2006-04-04&p-description=Desktop
```

runs report 120 in html format.

The #startdate# parameter (p-startdate) is set to 2006-04-04

and the #description# parameter (p-description) is set to "Desktop"

#### **Example 2: (chart)**

```
http://server_name:port/art/runReport?
reportId=121&p-date=2006-04-04
```

#### **Example 3: (group)**

```
http://server_name:port/art/runReport?
reportId=122&splitColumn=2&p-param=abc
```

runs report 122 in group on 2 columns mode (splitColumn parameter).

The parameter named param is set to "abc".

#### **Example 4: (pdf, public)**

```
http://server_name:port/art/runReport?
reportId=123&reportFormat=pdf&public=true
```

generates report as a pdf file without requiring the user to log in to ART

#### **Example 5: (dashboard)**

```
http://server_name:port/art/runReport?
reportId=124&p-param1=value1&p-param2=value2
```

runs report 124, setting values for parameters param1 and param2

#### **Example 6: (show selected parameters)**

```
http://server_name:port/art/runReport?
reportId=125&reportFormat=xls&p-startdate=2006-04-04&showSelectedParameters=true
```

generates report and includes the parameter value used in the report output

#### **Example 7: (multi-value parameters)**

```
http://server_name:port/art/runReport?
reportId=126&reportFormat=xls&p-category=Laptops&p-category=Desktops&p-category=Tablets
```

the #category# multi-value parameter (p-category) has 3 values defined:

Laptops, Desktops, Tablets

#### **Example 8: (default parameter values)**

```
http://server_name:port/art/runReport?
reportId=127
```

generates report using default parameter values

#### **Note:**

- If the report url contains some non-ASCII characters in the parameter values and they don't seem to be interpreted correctly, you may need to configure your application server to explicitly handle urls using UTF-8 encoding e.g. for Tomcat, edit the TOMCAT\_HOME\conf\server.xml file and add the **URIEncoding** attribute to the appropriate **Connector** element e.g.

```
<Connector port="8080" protocol="HTTP/1.1" URIEncoding="UTF-8"
...

```

## 10.5. Report Formats

When running reports interactively, or scheduling them for later execution, you can specify the format in which the results should be output. When running a report via url, you can specify the report format to be used using the `reportFormat` url parameter. Report format names are case sensitive. Some report formats are not available for certain types of reports.

Report Format	Display Name	Description
<b>html</b>	Browser	Shows results in a web page
<b>htmlGrid</b>	Browser (Grid)	Shows results in a web page. The data can be sorted by clicking on the column headers. Not available for scheduled jobs.
<b>htmlFancy</b>	Browser (Fancy)	Shows results in a web page, with minimal styling. Not available for scheduled jobs.
<b>htmlPlain</b>	Browser (Plain)	Shows results in a web page, with no styling.
<b>htmlDataTable</b>	Browser (DataTable)	Shows results in a web page. Data is displayed in pages. The data can be sorted by clicking on the column headers and one can filter or display certain rows by specifying some text to search for. Not available for scheduled jobs.
<b>xls</b>	Spreadsheet (xls)	Creates a Microsoft Excel spreadsheet file to download (xls format). Excel/OpenOffice/LibreOffice compatible
<b>xlsZip</b>	Spreadsheet (zip xls)	Creates a Microsoft Excel spreadsheet file to download (xls format), compressed using Zip.
<b>xlsx</b>	Spreadsheet (xlsx)	Creates a Microsoft Excel spreadsheet file to download (xlsx format). Excel/OpenOffice/LibreOffice compatible
<b>slk</b>	Spreadsheet (slk)	Creates a Microsoft Excel spreadsheet file to download (slk format). Excel/OpenOffice/LibreOffice compatible
<b>slkZip</b>	Spreadsheet (zip slk)	Creates a Microsoft Excel spreadsheet file to download (slk format), compressed using Zip.
<b>tsv</b>	File (text tsv)	Creates a tab-separated-values text file to download
<b>tsvZip</b>	File (zip tsv)	Creates a tab-separated-values text file to download, compressed using Zip

Report Format	Display Name	Description
<b>tsvGz</b>	File (gzip tsv)	Creates a tab-separated-values text file to download, compressed using GZip
<b>pdf</b>	Document (pdf)	Creates a pdf file to download
<b>png</b>	Image (png)	Creates a png file to download. Only available for charts.
<b>xml</b>		Creates xml output displayed on the browser
<b>rss20</b>		Output data as a rss2.0 compliant feed
<b>docx</b>	Document (docx)	Creates a Microsoft Word document file to download (docx format)
<b>odt</b>	Document (odt)	Creates an Open Document Format text document file to download (odt format)
<b>ods</b>	Spreadsheet (ods)	Creates an Open Document Format spreadsheet file to download (ods format)
<b>pptx</b>	Document (pptx)	Creates a Microsoft PowerPoint document file to download (pptx format). Only available for XDocReport - PPTX reports.
<b>json</b>	JSON	Outputs data in json format
<b>jsonBrowser</b>	JSON (Browser)	Outputs data in json format, for display in a browser. Data is output in a <code>&lt;pre&gt;</code> tag.
<b>csv</b>	CSV	Outputs data in csv format. Can also use another delimiter as defined in the report <b>Options</b> field - using the <b>delimiter</b> property. This format is primarily intended for jobs.

You can restrict which report formats are shown to users when running reports by modifying the Report Formats field in the Settings page.

**Note:**

- There is a limit set for the maximum number of rows that can be output with certain report formats. These limits are set, and can be modified from the Settings page.
- Use **xlsx**, **slk** or **tsv** report formats for large data sets. In particular, if you use the **xls** format for large data sets, you are likely to get out of memory errors. This is mainly due to the nature of the **xls** file format.

# 11. Multiple Statements

You may want to run some statements before or after the select statement for your report, e.g. to create a temporary table, create an index etc. Enter all your statements in the sql source section of the report, with each statement ending in a ; and specify which statement should be used as the report's results by setting the **Display Resultset** field of the report.

Display ResultSet	Description
0	The sql source doesn't contain multiple statements
1, 2, ... n	Use the specified statement, with the first statement being 1
-1	Use the select statement, regardless of how many statements exist
-2	Use the last statement, regardless of how many statements exist

If you set the Display Resultset to -2 to use the last statement, ensure that your database driver is at least JDBC 3.0 compliant.

Some RDBMSs may require extra configuration to allow for execution of multiple statements in a query, and some may not support it at all.

RDBMS	Comment
SQL Server	No extra configuration needed
PostgreSQL	No extra configuration needed
MySQL	Add the property <b>allowMultiQueries=true</b> to the jdbc url of the query's datasource e.g. <code>jdbc:mysql://localhost/mydb?allowMultiQueries=true</code>
CUBRID	Driver is JDBC 2.0 compliant so don't use the -2 option
Oracle	Not supported
HSQldb	Not supported

## 12. LOV Reports

LOV reports are used to generate parameter values for other reports. Dynamic LOV reports get their values from an sql query while static LOV reports use fixed values defined in the sql source section. An LOV report must have either one or two columns. The value of the first column is passed to the parameter. The value of the second column (if available) is displayed to the user. For example, for the following dynamic LOV query

```
SELECT location_id, location_name FROM LOCATIONS
```

Users see values from the `location_name` column while the actual parameter match is performed using values from the `location_id` column.

If the parameter options don't come from a database, you can use a static lov with something like the following in the sql source. If the parameter value and the display value are different, separate them with a |

```
local|Local  
international|Worldwide
```

### 12.1. Example

This section shows how to define a simple report to retrieve information about the reports stored in the ART database. The report has one parameter (the report name), obtained from a dynamic lov.

In order to proceed with this example you need to:

- Set up a datasource that points to the ART database
- Define a user and a report group

- **LOV Report**

This report is the one used to retrieve the list of values (i.e. the list of available reports that will be shown as a parameter). Use the **Configure | Reports** menu and then the **Add** button to create a new report. Set the name to "ART Report Names", the type to **LOV: Dynamic** and the following as the SQL source:

```
SELECT NAME FROM ART_QUERIES ORDER BY NAME
```

Select the ART database as the Datasource and **Save**.

- **Main Report**

This report retrieves the details of the reports that are defined in ART. It has one parameter whose values are retrieved from the "ART Report Names" report created above. Following the same process used for the previous report, name the report "ART Reports" and for the SQL source use:

```
SELECT NAME  
  , SHORT_DESCRIPTION  
  , DESCRIPTION  
  , UPDATE_DATE  
FROM ART_QUERIES WHERE NAME = #report_name#
```

Select the ART database as the Datasource and **Save**.



Use the **Configure | Parameters** menu and then the **Add** button to create a new single-value parameter named `report_name`. Select the **Use LOV** option and select "ART Report Names" in the **LOV Report** field. Click on the **Save** button to save the parameter details.

Use the **Configure | Reports** menu to go back to the Reports Configuration page. Find the main report, ART Reports, click on the **More | Parameters** button and **Add** a new parameter, setting the parameter to the `report_name` parameter that has just been created.

Use the **Configure | Access Rights** menu to define users who can run the main report. Now you can log in as the user and run the report.

**Note:**

- You don't have to assign access to the LOV report itself
- You can't use rules with static lovs. If you have a PostgreSQL database, you can create a dynamic lov that will return the static values, with syntax like the following

```
select id from
(select unnest(ARRAY[1, 5, 10]) as id) as id_list
where #rules#
```

## 13. Dynamic SQL

It is possible to create Dynamic SQL queries, allowing you to modify the structure of the query based on user parameters, possibly running different queries depending on the user input.

### 13.1. Using Groovy

You can use [groovy](#) in the SQL source section to achieve dynamic sql. The groovy script needs to return a string that has the sql to be executed. You need to define an option in the **Options** field of the report to specify that you are using groovy in the sql source. This option is named **usesGroovy** and needs to have the value **true** e.g.

```
{
  "usesGroovy": true
}
```

The groovy code is run through a sandbox and depending on the code in the script, you may need to specify additional classes in the **WEB-INF\groovy-whitelist.txt** file. If you get a security exception when running the report, add the indicated class to the groovy whitelist file and run the report again. You can also disable the sandbox completely by modifying the **WEB-INF\art-custom-settings.json** file and setting the **enableGroovySandbox** property to **false**. This modification will require an application restart to take effect.

#### 13.1.1. Example

A trivial example is given below.

```
def sql

//assumes the report has an integer parameter named id
if(id.value == 0) {
    sql = 'select id from users'
} else {
    sql = 'select name from users'
}
```

The demo database contains some sample reports that use groovy in the sql source to execute different queries based on parameter input.

### 13.2. Using XML tags

You can use xml tags in the SQL source section to achieve dynamic sql. The syntax is as follows. Tag names are case sensitive.

```
<IF>
<EXP1>value1</EXP1> <OP>operator</OP> <EXP2>value2</EXP2>
<TEXT> ... </TEXT>
<ELSETEXT> ... </ELSETEXT>
</IF>
```

ART parses the query and leaves only the text in the **<TEXT>** tag if the condition (value1 operator value2) is true or the text in the **<ELSETEXT>** tag if the condition is false. The EXP1 or EXP2 contents can be static values, single-value parameters or :tags, while the OP content is an operator (see

supported list below).

For example in the following query:

```
SELECT *
FROM <IF><EXP1>#level#</EXP1><OP>equals</OP><EXP2>summary</EXP2>
<TEXT> orders_summary </TEXT>
<ELSETEXT> orders_details </ELSETEXT>
</IF>
WHERE VENDOR like #vendor#
<IF><EXP1>#date#</EXP1><OP>is not null</OP>
<TEXT> AND order_date > #date# </TEXT>
</IF>
```

if the #level# parameter is set to "summary" and the #date# parameter is not null, ART rewrites the query as:

```
SELECT *
FROM orders_summary
WHERE VENDOR like #vendor#
AND order_date > #date#
```

if the #level# parameter is not set to "summary" and the #date# is null, ART rewrites the query as:

```
SELECT *
FROM orders_details
WHERE VENDOR like #vendor#
```

### **13.2.1. Operators**

Operator	Description
<b>eq</b> or <b>equals</b>	equals (case insensitive)
<b>neq</b> or <b>not equals</b>	not equals (case insensitive)
<b>ln</b>	less than (numbers)
<b>gn</b>	great than (numbers)
<b>la</b>	less than (alphabets) (case insensitive)
<b>ga</b>	great than (alphabets) (case insensitive)
<b>is blank</b> or <b>is null</b>	returns true if EXP1 is blank (EXP1 can never be the null object)
<b>is not blank</b> or <b>is not null</b>	returns true if EXP1 is not blank (EXP1 can never be the null object)
<b>starts with</b>	returns true if EXP1 string begins with EXP2 (case insensitive)
<b>ends with</b>	returns true if EXP1 string ends with EXP2 (case insensitive)
<b>contains</b>	returns true if EXP1 string contains EXP2 string within it (case insensitive)
<b>eq cs</b> or <b>equals cs</b>	equals (case sensitive)
<b>neq cs</b> or <b>not equals cs</b>	not equals (case sensitive)
<b>la cs</b>	less than (alphabets) (case sensitive)
<b>ga cs</b>	great than (alphabets) (case sensitive)
<b>starts with cs</b>	returns true if EXP1 string begins with EXP2 (case sensitive)
<b>ends with cs</b>	returns true if EXP1 string ends with EXP2 (case sensitive)
<b>contains cs</b>	returns true if EXP1 string contains EXP2 string within it (case sensitive)

### 13.2.2. Examples

**Dynamic OR/AND selection:**

```

SELECT *
FROM orders_summary
WHERE VENDOR like #vendor#
<IF><EXP1>#logic#</EXP1><OP>equals</OP><EXP2>OR</EXP2>
<TEXT> OR country = #country# </TEXT>
<ELSETEXT> AND country = #country# </ELSETEXT>
</IF>

```

**Dynamic ORDER BY:** The order by part is driven by the user input

```

SELECT *
FROM orders_summary
WHERE VENDOR like #vendor#
<IF><EXP1>#sortby#</EXP1><OP>equals</OP><EXP2>Vendor</EXP2>
<TEXT> ORDER BY 1 </TEXT>
<ELSETEXT> ORDER BY 2 </ELSETEXT>
</IF>

```

**Dynamic query selection:** A different query is executed depending on user input

```

<IF><EXP1>#showaddr#</EXP1><OP>equals</OP><EXP2>Y</EXP2>
<TEXT>
SELECT name, code, address, phone FROM VENDOR
WHERE VENDOR like #vendor#
</TEXT>
<ELSETEXT>
SELECT name, code, vat, pay_term FROM VENDOR
WHERE VENDOR like #vendor#
</ELSETEXT>
</IF>

```

**Dynamic SQL with tags:** The condition is verified only if the date supplied by the user (#date#) is earlier than the system date at report execution time (:DATE tag)

```

SELECT *
FROM orders_summary
WHERE VENDOR like #vendor#
<IF><EXP1>#date#</EXP1><OP>la</OP><EXP2>:DATE</EXP2>
<TEXT> AND order_date > #date# </TEXT>
</IF>

```

**Check user input:** Show a warning instead of executing the query

```

<IF><EXP1>#value#</EXP1><OP>ln</OP><EXP2>10000</EXP2>
<TEXT> SELECT ... </TEXT>
<ELSETEXT> SELECT 'Value too High' "Warning" </ELSETEXT>
</IF>

```

the select statement to use to show the warning depends on the DBMS (for example in Oracle you should use SELECT 'Value too High' "Warning" FROM DUAL)

**Dynamic WHERE condition:** E.g. to drill down on null values

```

SELECT * FROM customers
WHERE
<IF><EXP1>#email#</EXP1><OP>equals</OP><EXP2>null</EXP2>
<TEXT>customer_email is null</TEXT>
<ELSETEXT>customer_email=#email#</ELSETEXT>
</IF>

```

## 14. Drill Down Reports

This functionality provides a main report with links from which a user can click to get to a different (drill down) report. For example, the main report can display summary information and a user can click on the drill down report link to display detailed information about a particular item.

The main report can be a normal tabular query - displayed in one of the html view modes - or a chart. The drill down report can be of any type.

### 14.1. Defining the main report

The main report is created like any other report. Once the report is saved, use the **More | Drilldowns** button to add or modify the drill down reports that will be available for that report.

### 14.2. Defining the drill down report

The drill down report is created like any other report. The only requirement is that it needs to have at least one single-value parameter defined, with this parameter having the **Drilldown Column Index** field with a value greater than or equal to 1. The Drill Down Column refers to the column in the main report that will provide the value for the parameter. If the value will come from the first field, the drill down column will have a value of 1 and so on.

### 14.3. Example

- **Sales Summary** (Main report)

```
select REGION "Region", SUM(VOLUME) "Volume"
from ORDERS
where ...
```

- **Sales Details** (Drill Down report)

```
select REGION, CITY, ITEM, VOLUME
from ORDER_DETAILS
WHERE REGION = #region#
```

On the Drill Down report, when defining the #region# parameter, set the option **Drilldown Column Index** to 1. This means this parameter will match with the first column value on the Main Report i.e. "Region"

From now on this report will appear as an available Drill Down report.

On the Main report, select the **More | Drilldowns** option

Select **Add** and:

- Pick the Drill Down report from the list
- Specify the link column title (header) and link text
- Select which report format will be used to display the Drill Down Report results. Using the **All** option will enable the user to select the report format.
- Set if the result of the drill down report should be opened in a new window

When running the Main Report, a new column will appear on the right:

Click on it and the drill down report will be executed - the #region# parameter will match with the first column value

## 14.4. Using the main report's parameters

In addition to using the main report's column values, a drill down report can also use the main report's parameter values. If in our example above the Main Report allowed the user to select a city, e.g.

```
select REGION "Region", SUM(VOLUME) "Volume"
from ORDERS
WHERE CITY=#city# AND ...
```

The Drill Down Report can make use of this #city# parameter even if it is not among the columns displayed by the Main Report's select statement. To use the value of the #city# parameter in the Drill Down Report,

- Define the parameter with the same name as in the Main Report
- Set the Drilldown Column Index for this parameter to 0, since the value of this parameter will not be coming from the columns displayed in the Main Report

So the Drill Down Report would look something like

```
select REGION, CITY, ITEM, VOLUME
from ORDER_DETAILS
WHERE REGION = #region# and CITY=#city#
```

Here the #region# parameter will still be set as Drilldown Column 1, since it's getting its value from the Main Report's column 1, while the #city# parameter will be set as Drilldown Column 0, since it's not getting its value from any of the Main Report's columns (the Main Report's select doesn't include the city column).

The parameter values from one report are passed down to all drill down reports down the line. So if this drill down report had another drill down report, that report can also use the value of the #city# parameter set in the Main Report.

### Note:

- Multiple Drill Down Reports can be linked to one Main Report. Extra columns will appear with links allowing the user to select which drill down report to run.
- A Drill Down Report can have several parameters to match any number of column values from the Main Report. It can also have parameters that use values from the Main Report's parameters.
- If the Main Report is a chart, the drill down parameter will get the following values
  - Drilldown column 1 = data value
  - Drilldown column 2 = category name
  - Drilldown column 3 = series name

## 15. Dynamic Datasources

When you define a report, you specify the datasource from which data will be retrieved. Sometimes, you may have several databases that have the same schema but contain different data e.g. a live production database, a test database, a database that has data as at end of month etc. You may want to have the possibility of running the same query over these different databases. Rather than creating several reports with identical sql but different datasources, you can create one report and have the datasource as a parameter that can be selected at runtime. This eliminates the work of creating multiple reports and makes management of the relevant sql much easier - if you need to change the sql, you only have to do it in one place. The process of using dynamic datasources is as follows.

- Create a datasource for each of your target databases
- Create the report that will retrieve the data you want. What you select in the Datasource field is not important as the report will use a dynamically specified datasource. It is advisable though to set this to a valid, default datasource for your sql.
- Define a single-value parameter for your report, setting the Data Type to Datasource. If you would like the available datasources to be selected from a list, create and specify the LOV report that will display the desired datasources.

When you run the report, it will use the value of the datasource parameter to determine which datasource to run the report on. The value of this parameter should be an existing datasource id.



## 16. RSS Feeds

Since ART 1.11 it is possible to create RSS 2.0 compliant feeds out of any datasource.

This is achieved in three steps:

- Create a report where column names follow RSS item element naming convention - see below. The resultset column names are used as the names of the <item> sub-elements in the produced xml file.
- Grant access to **public\_user**
- Run the report via direct URL, setting the **reportFormat** parameter to **rss20**. This URL, when executed via a browser or RSS reader/aggregator will return an RSS feed.

As a minimum, the report should contain a column named **title** or **description**.

Element Name	Description
<b>title</b>	The item title
<b>description</b>	The item description
<b>pubDate</b>	The date where the specific item is published
<b>link</b>	The URL pointing to the items
<b>guid</b>	The unique id of the item. If present, a feed aggregator may choose to use this string to determine if an item is new. It may/should be a URL pointing to the full item.

Please refer to the RSS specification for more details.

For example the following query:

```
select col1 "title", col2 "description" , col3 "pubDate", col4 "guid"
from myTable
```

Will produce the following RSS 2.0 XML content:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
  <title>[Report Name]</title>
  <link>[As configured in ART settings]</link>
  <description>[Report Name]</description>
  <pubDate>[Current date]</pubDate>
  <generator>http://art.sourceforge.net</generator>
  <item>
    <title>[resultset "title" column]</title>
    <description>[resultset "description" column]</description>
    <guid>[resultset "guid" column]</guid>
    <pubDate>[resultset "pubDate" column]</pubDate>
  </item>
  <item>
    ...
```

```
</item>
...
</channel>
</rss>
```

**Note:**

- Date/Timestamp columns are converted to RfC 822 formatted strings. Characters <, > and & are converted to html escape entities. If needed, for any other special characters, you should take care of proper conversion.
- Only the <item> elements are customizable by using the proper column names. Channel elements are as in the example: the default channel link tag can be set by an administrator from the Settings page.

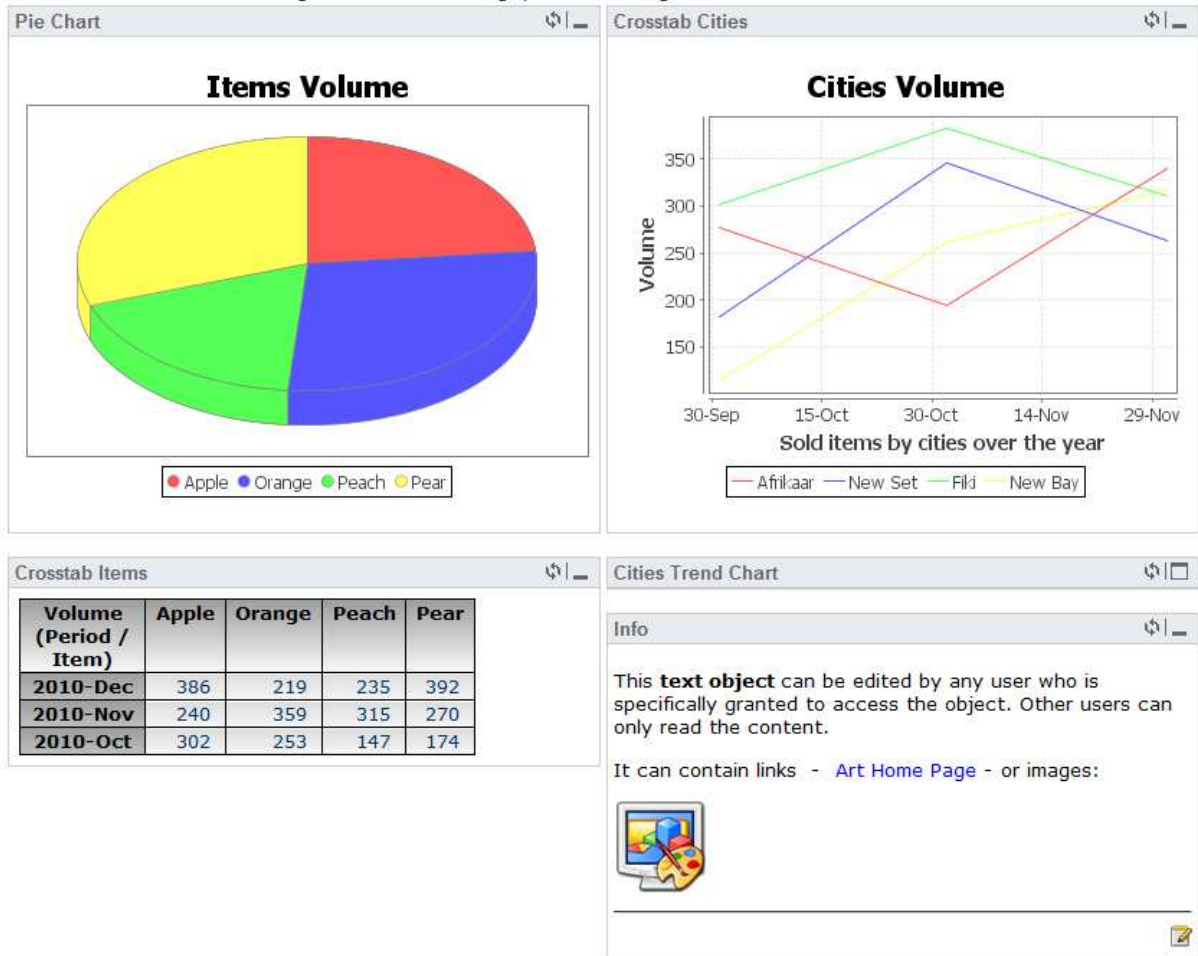
## 17. Dashboards

Dashboards are used to display reports on a single portal-like page.

Any ART report can be considered a portlet i.e. a small embeddable frame within a web page. Each portlet uses AJAX components to dynamically load itself within the page. Users can refresh or minimize each portlet independently by clicking on the buttons at the top of the portlet frame. When a user runs a dashboard, he can choose to modify the default parameters used by the reports in the dashboard. All the embedded reports are parsed and their parameters, if any, are displayed. If several reports use the same parameter, it is displayed only once.

## Regional Scorecard

Shows a dashboard collecting some of the existing queries in a single screen



## 17.1. Regular Dashboards

### 17.1.1. Features

- You can define as many columns as you need (tag: <COLUMN>)
- For each column you can specify a size (tag: <SIZE>)
- Within each column you can specify one or more portlets (tag: <PORTLET>)
- Each portlet has a title (tag: <TITLE>)
- A portlet can display any ART report using the report's ID (tag: <REPORTID>)
- An external URL (tag: <URL>) can also be displayed within a portlet
- A portlet can be refreshed by the end user by clicking on the refresh button. You can set a refresh time (in seconds) to have the portlet auto-refresh itself. (tag: <REFRESH>)
- Each portlet is rendered on page load. You can override this and let the user decide when he wants to see the content (by clicking on the refresh button). (tag: <ONLOAD>)

## 17.1.2. Example Syntax

XML-like syntax is used to define the dashboard structure

```
<DASHBOARD>

  <COLUMN>
    <!-- column size: auto|small|medium|large. default is auto-->
    <SIZE>medium</SIZE>

    <!-- create a new portlet within this column
    to embed an ART report (tabular, chart, text, etc) -->
    <PORTLET>
      <TITLE>Portlet title</TITLE>

      <!-- (optional, default is true) load content when page is displayed -->
      <ONLOAD>>false</ONLOAD>

      <!-- (optional, default is -1 meaning never) refresh content every 30 seconds-->
      <REFRESH>30</REFRESH>

      <!-- embed ART report -->
      <REPORTID>2</REPORTID>
    </PORTLET>

    <!-- create a new portlet within this column
    to embed an external html page -->
    <PORTLET>
      <TITLE>Portlet title</TITLE>
      <URL>http://my.site.com/page.html</URL>
    </PORTLET>

    <!-- .. you can add as many portlets as you want -->
  </COLUMN>

  <COLUMN>
    <!-- you can add as many columns as you want -->
  </COLUMN>

</DASHBOARD>
```

### Note:

- Using the <REPORTID> tag, tabular reports are displayed using the htmlFancy format. To override this, use the <URL> tag and type the direct access URL for the report
- Tag names are case sensitive (use uppercase)
- Minimum refresh period allowed is 5 seconds. Be careful with this so as not to overload the server and databases.
- If you need to use xml special characters within tag content, ensure to replace them with the relevant xml character entities e.g. if the <URL> tag should contain a url like `http://localhost:8080/art/runReport?reportId=1&p-param1=test`, you'll need to replace the & character with `&amp;`, to have the url as `http://localhost:8080/art/runReport?reportId=1&amp;p-param1=test`. For a list of the 5 special xml characters and their respective replacement strings or character entities, see <http://support.esri.com/technical-article/000005870>
- For pdf output of dashboards, only tabular and standard chart reports will be included in the pdf output.

## 17.2. Gridstack Dashboards

Gridstack dashboards allow for specifying exact positioning of report items within the dashboard, specifying the height and width that the report items should occupy, allowing for resizing the items dynamically using the mouse, and moving the items around using drag-and-drop.

### 17.2.1. Dashboard Properties

Gridstack dashboards also use xml syntax, similar to regular dashboards. A number of tags are used to define properties of the dashboard.

Tag	Description
<DASHBOARDWIDTH>	(optional - integer) default = 12. The number of columns to be taken up by the grid, forming the basis of the horizontal or x axis of the grid co-ordinates.
<FLOAT>	(optional - boolean) default = false. Allows for moving of report items to arbitrary locations on the dashboard. If false, grid items can only snap to given locations.
<ANIMATE>	(optional - boolean) default = false. Allows for using some animation when grid items are moved around.
<DISABLEDRAG>	(optional - boolean) default = false. Disallows dragging of items.
<DISABLERESIZE>	(optional - boolean) default = false. Disallows resizing of items.
<CELLHEIGHT>	(optional - string) default = 60px. Defines the dimensions of one unit of the grid co-ordinates on the vertical or y axis. Can also be defined in terms of <b>em</b> and <b>rem</b> units instead of pixels e.g. 10em or 10rem. Can also be defined as " <b>auto</b> " in which case the height will be calculated from the cell width.
<VERTICALMARGIN>	(optional - string) default = 20px. Defines the size of the vertical gap between cells. Can also be defined in terms of <b>em</b> and <b>rem</b> units instead of pixels e.g. 2em or 2rem.
<ALWAYSSHOWRESIZEHANDLE>	(optional - boolean) default = false. Determines if the resizing handles on the bottom corners of a cell are shown even if the user is not hovering over the item.
<DASHBOARDHEIGHT>	(optional - integer) default = 0. The maximum number of rows or y units on the grid co-ordinate system. Items cannot be resized or moved beyond these limits. A value of 0 means that there is no maximum.

## 17.2.2. Item Properties

A gridstack dashboard has a number of items, enclosed within <ITEM> tags. These items have a number of properties as specified in the following tags.

Tag	Description
<XPOSITION>	( <b>required</b> - integer) The x axis position of the item.
<YPOSITION>	( <b>required</b> - integer) The y axis position of the item.
<WIDTH>	( <b>required</b> - integer) The width of the item. This defines the number of columns that the item will occupy, in relation to the number of columns of the whole grid as defined by the dashboard's <DASHBOARDWIDTH> property.
<HEIGHT>	( <b>required</b> - integer) The height of the item. This defines the number of rows that the item will occupy, one row being the size defined in the dashboard's <CELLHEIGHT> property.
<NORESIZE>	(optional - boolean) default = false. Disables element resizing.
<NOMOVE>	(optional - boolean) default = false. Disables element moving.
<AUTOPOSITION>	(optional - boolean) default = false. Specifies to ignore the <XPOSITION> and <YPOSITION> and instead place the element in the first available position.
<LOCKED>	(optional - boolean) default = false. Defines that the widget will be locked. It means another widget wouldn't be able to move it during dragging or resizing. The widget itself can still be dragged or resized. You need to specify the <NORESIZE> and <NOMOVE> options to completely lock the widget.
<MINWIDTH>	(optional - integer) default = 0. Specifies the minimum width beyond which the item cannot be resized.
<MINHEIGHT>	(optional - integer) default = 0. Specifies the minimum height beyond which the item cannot be resized.
<MAXWIDTH>	(optional - integer) default = 0. Specifies the maximum width beyond which the item cannot be resized. A value of 0 means no maximum.
<MAXHEIGHT>	(optional - integer) default = 0. Specifies the maximum height beyond which the item cannot be resized. A value of 0 means no maximum.

In addition to these properties that are unique to gridstack items, the following properties of regular dashboard portlets should also be used within gridstack items, with the same effect: <TITLE>, <REPORTID>, <URL>, <REFRESH>, <ONLOAD>.

## 17.2.3. Example Syntax

```
<DASHBOARD>

  <ITEM>
    <TITLE>Report 1</TITLE>
    <REPORTID>1</REPORTID>
```

```

        <XPOSITION>0</XPOSITION>
        <YPOSITION>0</YPOSITION>
        <WIDTH>4</WIDTH>
        <HEIGHT>4</HEIGHT>
    </ITEM>

    <ITEM>
        <TITLE>Report 2</TITLE>
        <REPORTID>2</REPORTID>
        <XPOSITION>8</XPOSITION>
        <YPOSITION>0</YPOSITION>
        <WIDTH>4</WIDTH>
        <HEIGHT>4</HEIGHT>
    </ITEM>

</DASHBOARD>

```

## 17.3. Tabbed Dashboards

Both regular and gridstack dashboards can be modified to have their components displayed within tabs. This would be done by adding the optional `<TABLIST>` tag to the dashboard xml. Tabs will then be defined according to the desired number of tabs, and which portlets/items should appear in which tabs.

### 17.3.1. Example Syntax

```

<DASHBOARD>

    <ITEM>
        <TITLE>Report 1</TITLE>
        <REPORTID>1</REPORTID>
        <XPOSITION>0</XPOSITION>
        <YPOSITION>0</YPOSITION>
        <WIDTH>4</WIDTH>
        <HEIGHT>4</HEIGHT>
    </ITEM>

    <ITEM>
        <TITLE>Report 2</TITLE>
        <REPORTID>2</REPORTID>
        <XPOSITION>8</XPOSITION>
        <YPOSITION>0</YPOSITION>
        <WIDTH>4</WIDTH>
        <HEIGHT>4</HEIGHT>
    </ITEM>

    <TABLIST>
        <DEFAULTTAB>1</DEFAULTTAB>
        <TAB>
            <TITLE>Tab 1</TITLE>
            <ITEM>2</ITEM>
        </TAB>
    </TABLIST>

</DASHBOARD>

```



### 17.3.2. Tag Descriptions

Tag	Description
<TABLIST>	Container tag for tab definitions. If present, only the items defined in the tabs will be displayed. If absent, all items as defined in the main dashboard xml will be displayed.
<DEFAULTTAB>	(optional - integer) default = 1. Defines the index of the tab that will be active when the dashboard is displayed. The first tab has an index of 1.
<TAB>	Container tag for a specific tab. The <TABLIST> can have any number of <TAB> tags.
<TITLE>	(optional - string) default = blank/empty string. Defines the title/heading/name of the tab. Even though this tag is optional, it is recommended to always set it so as to have a better visual indication for users of the different tabs.
<ITEM>	(optional - integer) no default. The index of the item to display, as it appears in the main dashboard xml definition. The dashboard portlets/items are assigned an index according to the order in which they appear in the xml, with the first item having an index of 1. So setting this to 1 indicates that the first portlet/item in the main dashboard xml should be displayed. A tab can have any number of <ITEM> tags.

## 18. jXLS Reports

ART uses the [jXLS](#) library to provide support for reports based on pre-formatted MS Excel spreadsheets. For details on the syntax of these jXLS templates, see the documentation available on the jXLS website. Generally, you'll follow the following steps to create a jXLS report.

- Create an Excel file (either xls or xlsx) with any text and formatting that will be required in the final report
- Insert jXLS syntax as appropriate to define where the raw data should go
- Create a new report in ART and select the jXLS template to use
- If the report is of type **jXLS: ART Query**, the sql used to retrieve the data will be the one defined in the SQL source section. You can use parameters, rules and any other features of ART reports. In the template, use the identifier **results** to output report values. e.g.

```
jx:each(items="results" var="row" lastCell="E10")
```

```
${row.description}
```

- If the report is of type **jXLS: Template Query**, the sql used to retrieve the data will be the one defined in the template. The database used for the report will be the one selected in the datasource field of the report.
- If the report uses parameters, avoid using the following names for the parameters as these are used for special purpose variables i.e. `results`, `jdbc`, `conn`, `params`, `locale`.

## 19. JPivot Reports

There are 3 report types that use the [JPivot](#) library to provide OLAP functionality.

### 19.1. Mondrian

This report type uses ART's internal [mondrian](#) engine to run OLAP queries. To use it, do the following.

#### 19.1.1. *Create the schema xml file*

- Create a mondrian (version 3) cube schema file that defines your cube

#### 19.1.2. *Create a datasource*

- Create a new datasource in ART using the **Configure | Datasources** menu and then clicking on the **Add** button
- In the **Datasource Type** section, select **OLAP**
- Provide the url, username and password to use to access the database that has the star schema tables referenced in the schema xml file
- Save the datasource

#### 19.1.3. *Create the report*

- Create a new report in ART of the type **JPivot: Mondrian**
- In the **Datasource** field, select the datasource you have just created
- For the **Template** field, select the schema file for the cube
- In the MDX source field, enter the MDX for your report

#### 19.1.4. *Roles*

If you need to define roles that users should have when accessing the cube, create a rule for the report and assign users rule values for this rule according to the required roles. The name of the rule doesn't matter. You can also use a dummy report column name when defining the report rule. Roles are case-sensitive as per their appearance in the schema xml file so rule values should be exactly as they appear in the schema file. If you specify a rule value that doesn't exist in the schema file, the report will have an error and won't run. If you need to specify multiple roles, specify multiple rule values.

### 19.2. Mondrian XMLA

This report type is used to access cubes defined in an external mondrian server. To use it, do the following.

### 19.2.1. Create the schema xml file

- Create a mondrian (version 3) cube schema file that defines your cube

### 19.2.2. Create the datasources.xml file

- Create the datasources.xml file that has the cube details. ART can act as a Mondrian XMLA server by configuring a file named **datasources.xml** and putting it in the **ART\_HOME\WEB-INF\work\templates** directory, and using a url like <http://server-name:8080/art/xmla> as the XMLA url.

### 19.2.3. Create a datasource

- Create a new datasource in ART using the **Configure | Datasources** menu and then clicking on the **Add** button
- In the **Datasource Type** section, select **OLAP**
- In the **URL** field, enter the url to use to access the xmla server e.g. <http://localhost:8080/art/xmla>
- If the AuthenticationMode section of the datasources.xml file is set to Unauthenticated, leave the **Username** and **Password** fields blank. Otherwise enter the username and password to be used to access to the cube.
- Save the datasource

### 19.2.4. Create the report

- Create a new report in ART of the type **JPivot: Mondrian XMLA**
- In the **Datasource** field, select the datasource you have just created
- Enter the required **XMLA Datasource** name. This should match exactly with the contents of the **DataSourceName** in the **datasources.xml** file defined on the external server. These names can be any string and don't have to match any name in the schema xml file.
- Enter the required **XMLA Catalog** name. This should match exactly with the contents of the **Catalog** name in the **datasources.xml** file defined on the external server. These names can be any string and don't have to match any name in the schema xml file.
- In the MDX source field, enter the MDX for your report

### 19.2.5. Potential errors

- If you encounter a **404** error, the URL specified in the ART datasource may be invalid.
- If you encounter a **No metadata for catalog** error, the catalog name in the **XMLA Catalog** field may be misspelt. Catalog names are case sensitive.
- If you get a **XMLA connection datasource not found** error, the datasource name in the **XMLA Datasource** field may be misspelt. Datasource names are case sensitive.
- If you get a **XMLA Discover unparse results error** error, the file specified in the **Definition** section of the **Catalog** section of the **datasources.xml** may not exist. Alternatively, the JDBC details provided in the **DataSourceInfo** tag may not be correct. Alternatively, the **Catalog** name may not be matching the **name** attribute of the **Schema** tag in the cube's schema xml file. Alternatively, the **DataSource** name may not be matching the **XMLA Datasource** field of the report.
- If you get a **No metadata schema dimensions for catalog** error, the catalog name entered in the **XMLA Catalog** field may be incorrect. Note that if a change is made to the datasources.xml, the

application needs to be restarted for the changes to take effect.

## 19.2.6. Accessing the Mondrian XMLA instance from MS Excel

You can configure ART to serve as a mondrian XMLA server by defining a file named **data-sources.xml** and placing it in the **ART\_HOME\WEB-INF\work\templates** directory. A sample data-sources.xml file can be found in the **ART\_HOME\WEB-INF\etc** directory. The steps outlined here refer to MS Excel 2010 and may be different for different versions of Excel.

- Download the XMLA connect provider from <https://sourceforge.net/projects/xmlaconnect/files/>
- Run the .exe file
- Open Excel
- Select the **Data** tab
- Click on the **From Other Sources** menu and then select **From Data Connection Wizard**
- In the dialog that opens, select **Other/Advanced** and then click on **Next**
- Select the **XMLA Data Source** provider and click on **Next**
- In the **Location** box, type a url like <http://localhost:8080/art/xmla> as per the ART installation
- Click on the drop-down arrow in the **Catalog** box and select the required item
- Click on **OK**
- In the next screen, select which cube you want to deal with and click **Next**
- Click on **Finish** to complete the process
- Click on **OK** to have the pivot table details loaded in the current work sheet
- From the **Pivot Table Field List** on the right, drag a measure under the Values section to the Values section in the below panel
- You should now have a value in your worksheet displaying the total value for this measure
- You can drag items to the Row Labels, Column Labels and Report Filter sections in order to do further analysis
- You can also slice the data. Click on any cell in the pivot table. From the **Options** tab, click on the **Insert Slicer** menu, check an item and click on **OK**. You can now slice your data by clicking the different available values of the slicer you selected, with the data being updated to reflect your selection. To remove the slicer, right-click on its heading or any empty space within the slicer and click on the **Remove ...** menu.

If you want to view the MDX that is generated for the analysis you are doing, download and install the **OLAP Pivot Table Extensions** plugin for Excel from <https://olappivottableextend.codeplex.com/>. Once you download the correct version of the plugin for your version of Excel, and start Excel, you can view the MDX for any analysis by right-clicking on any cell in the pivot table data, selecting the **OLAP Pivot Tables Extensions** menu and selecting the **MDX** tab.

## 19.3. SQL Server XMLA

This report type is used to access cubes defined in a Microsoft SQL Server Analysis Services (SSAS) server. Before being able to run reports based on a SSAS cube, you need to configure SSAS for HTTP access.

### 19.3.1. Configure SSAS for HTTP access

Some resources on how to do this are as follows

- <http://geekswithblogs.net/darengosbell/archive/2007/11/04/SSAS-Connecting-via-HTTP-on-Windows-Vista.aspx>
- [http://ssas-wiki.com/Articles#HTTP\\_Access](http://ssas-wiki.com/Articles#HTTP_Access)

After you have finished the configuration, start IIS.

To test that you can access SSAS via HTTP, open SQL Server Management Studio, select Analysis Services as the Server type, put the URL in the Server name box e.g. <http://localhost/olap/msmdpump.dll> and try to connect. If you are able to connect, access to the server is fine, and next you need to provide access rights to the cube.

### 19.3.2. Provide access to the cube

These steps outline the process for providing access on SQL Server 2008R2 and may be different for other versions of SQL Server.

- Using SQL Server Management Studio, connect to the SSAS server using an administrator user
- Expand the **Databases** folder then expand the desired database
- Right-click on the **Roles** section for the database and select the **New Role** menu to create a new role
- Provide a **Role name** for the role and tick the **Read definition** box to give the role read access
- Select the **Membership** section on the left-hand pane to add the users who will have access. If HTTP access is configured for anonymous access, add the **IUSR** user. If not, add the user whose credentials will be used to access the cube. Enter a given user and click on the **Check Names** button to have the user retrieved.
- Select the **Data Sources** section on the left-hand pane and then select **Read** in the **Access** column of the desired cube
- Select the **Cubes** section on the left-hand pane and the select **Read** in the **Access** column for the desired cube
- Click on **OK** to save the role

To verify that you can access the cube and generate reports from it

- Open a new SQL Server Management Studio instance and connect to SSAS using the msmdpump URL e.g. <http://localhost/olap/msmdpump.dll> and using the user which you assigned to the role.
- Expand the **Databases** folder
- Expand the folder for your database
- Expand the **Cubes** folder
- Right click on your cube and select **Browse**
- You should be able to drag and drop dimension members to the analysis pane to display some data

### 19.3.3. Create a datasource

- Create a new datasource in ART using the **Configure | Datasources** menu and then clicking on the **Add** button
- In the **Datasource Type** section, select **OLAP**
- In the **URL** field, enter the url to use to access SSAS e.g. <http://localhost/olap/msmdpump.dll>
- If the SSAS server is configured to allow anonymous access, leave the **Username** and **Password** fields blank. If it's configured to require basic authentication, enter the username and password of a user in a role that has access to the required cube.
- Save the datasource

### 19.3.4. Create the report

- Create a new report in ART using the **Configure | Report** menu and then clicking on the **Add** button
- In the **Type** field, select **JPivot: SQL Server XMLA**
- In the **Datasource** field, select the datasource you have just created
- In the **XMLA Catalog** field, enter the database name for your database on the SSAS server
- In the MDX source field, enter the MDX for your report

You should now be able to run the report and perform analysis. If you encounter a **connection refused** error, IIS may not be started. If you encounter a **No metadata schema dimensions for catalog** error, you may not have permissions on the cube.

#### Note:

- You can use single-value parameters in the mdx using the direct parameter substitution syntax `#!<param name>#` e.g.

```
SELECT
{ [Measures].[Sales Amount], [Measures].[Tax Amount] } ON COLUMNS,
{ [Date].[Fiscal].[Fiscal Year].&{#!FromYear#}, [Date].[Fiscal].[Fiscal Year].&{#!ToYear#} } ON ROWS
FROM [Adventure Works]
WHERE ( [Sales Territory].[#!Territory#] )
```

- You can also use multi-value parameters. In this case, construct the LOV query such that it returns strings as they would appear in the MDX. e.g.

Example LOV

```
SELECT CONCAT("[Order.Order Sale Type].[",sale_type_code,"]") as typecodes, sale_type_code
FROM dim_orders
```

Use multi-value parameter in MDX as desired...

```
SELECT ... FROM ...
WHERE {#!saleTypes#}
```

- JPivot reports use cached data where possible. To manually clear the cache, use the **Configure | Caches** menu and Clear the **JPivot** cache. If you'd like the cache to be automatically cleared periodically, use the **Mondrian Cache Expiry Period** setting on the **Settings** page.

## 20. Rules

Rules are used to filter report results. They allow the same report to be filtered depending on the user that is running it.

The following steps need to be performed in order to use rules:

- **Create the rule**

Use the **Configure | Rules** menu and then the **Add** button to specify the rule name

- **Link the rule to a report**

On the report definition page, select the **Uses Rules** option

In the SQL source section, use the special, hard coded placeholder **#rules#** where you want the rule values to go e.g

```
SELECT * from transactions
where #rules#
```

On the reports configuration page, find the report and use the **More | Rules** menu to specify which column the rule values will apply to

- **Assign rule values to users or user groups**

Use the **Configure | Rule Values** menu to assign rule values to users or user groups. Select the user or user group and rule name, and specify the rule values that will apply to that user or user group. If you want the user or user group to have multiple rule values, add each value separately. i.e enter the first value, then click on Add, then enter the next value etc.

### Example:

- You have a table named "employees" with the a column named "region"
- Create a rule named "GeoArea"
- Set up a report that selects rows from the employees table and link it to the rule named "GeoArea" on the column "employees.region"
- Link the user to the rule "GeoArea" for values NORTH and EAST

When the user runs the report, he will extract only the rows where the "region" column has the values NORTH or EAST (i.e. the SQL query will be modified on the fly before execution by adding AND employees.region IN ( 'NORTH' , 'EAST' ) to the WHERE clause).

### Note:

- If a report is linked to a rule but the user who runs the report has not been set up for that rule, the report execution will not be performed and an error message will be displayed.
- If the report uses multiple rules, each user that needs to execute it must have values set for all the rules that the report uses.
- If the report uses parameters, the placeholder **#rules#** shouldn't be used for any parameter as it will conflict with the special placeholder for rules.
- If you don't want results filtered for a given user e.g. an administrator, manager etc, define a rule value for this user where the rule value is **ALL\_ITEMS**.



## 21. Chained Parameters

A chained parameter is one whose values depend on the value selected in another parameter. The parameter that triggers the change is called the "parent". A chained parameter can be used as a parent for another parameter, which in turn can have another child parameter and so on.

### 21.1. Example

Suppose we want to view customers who reside in a particular city. We want the user to select a country and then have cities in that country displayed. He'll then pick one of these cities and run the report. We have a CUSTOMERS table, and one of the columns it has is CITY\_ID. We also have a COUNTRIES table which has COUNTRY\_ID and COUNTRY\_NAME columns. Finally, we have a CITIES table with CITY\_ID, CITY\_NAME and COUNTRY\_ID columns.

We could take the following steps to set up the report.

- Create a Dynamic LOV report named **countries** that will display country names. The country will be the parent parameter. Depending on what country is selected, the available cities will change. The report would have the following SQL.

```
SELECT COUNTRY_ID, COUNTRY_NAME
FROM COUNTRIES
ORDER BY 2
```

- Create a Parameter named **countryId** with a Data Type of **Integer**, set it to **Use LOV** and select "countries" as the **LOV Report**.
- Create a Dynamic LOV report named **cities** that will display city names. The query needs to have a parameter label for the country parameter. This will be replaced at runtime with the country selected.

```
SELECT CITY_ID, CITY_NAME
FROM CITIES
WHERE COUNTRY_ID = #countryId#
ORDER BY 2
```

- Once you have saved the cities report, select the **More | Parameters** option for this report to allocate the countryId parameter.
- Add a Report Parameter and set the **Parameter** field to the **countryId** parameter created earlier.
- Create a Parameter named **cityId** with a Data Type of **Integer**.
- Create the main report. The SQL for the report will need to have a placeholder for the cityId parameter.

```
SELECT FIRST_NAME, LAST_NAME, EMAIL
FROM CUSTOMERS
WHERE CITY_ID = #cityId#
```

- Once you have saved the main report, select the **More | Parameters** option for this report in order to allocate the report parameters.

- Add a Report Parameter and set the **Parameter** field to the **countryId** parameter created earlier.
- Add another Report Parameter, set the **Parameter** field to **cityId** and the **Chained Parents** field to "countryId" i.e. the name of the parent parameter for this chained parameter.

Now when you run the main report, you are presented with two drop down boxes. Depending on what country you select, a different set of cities is displayed, from which you can choose one and run the report to see the final results.

## 22. Scheduling

In order to schedule a report, a user needs an access level of at least **Schedule User**. Once logged in, a user's scheduled reports are available from the **Jobs** menu.

### 22.1. Scheduling a new job

Take the following steps to schedule a new job

- From the Reports page, select the report to schedule and click on the **Schedule** button
- Specify the job details and Save
- The job will now run as per the schedule defined

#### Note:

- Not all report types can be scheduled
- The **Runs To Archive** field indicates, for Publish and Conditional Publish jobs, how many runs should be kept for viewing from the Archives page e.g. setting it to 5 will retain output from the last 5 runs in the Archives page. This allows users to access past job output as per their requirements.
- The **Fixed File Name** field indicates that the output file generated should use the file name indicated in this field e.g. `test.html`
- The **Batch File** field indicates the name of a batch file or script that should be run after the job completes. The batch file must exist in the **ART\_HOME\WEB-INF\work\batch** directory. You only set the file name in this field e.g. `test.bat` or `test.sh`. ART passes the file name of the generated output as the first parameter to the batch file so you can use this e.g. to copy the file to another location e.g. the `test.bat` file may have the following contents.

```
copy ..\export\jobs\%1 C:\temp
```

### 22.2. Job Types

- **E-Mail Output (Attachment)**  
The output is emailed as an attachment to the specified recipients (in the "To" section). The attachment type can be selected in the Output Format field.
- **E-Mail Output (Inline)**  
The output is emailed to the specified recipients, in the email body.
- **Publish**  
The output is saved to a file. The file is reachable from the Jobs page. Once the file is generated, a notification email is sent to the specified recipients (in the "To" section). Leave the "To" area empty if you don't want a notification email to be sent.
- **Alert**  
Send an email to the specified recipients if the first column of the first row in the query result has a value greater than zero. You can construct a simple SQL query to return a non zero value in case of a certain business condition.

Example query to be used for an alert. Send an email if a big order is made

```
SELECT count(*)
FROM todays_order_details
WHERE quantity>100
```

- **Just Run It**  
Simply run the report. Useful to launch stored procedures or perform periodic data updates.
- **Conditional E-Mail Output (Attachment)**  
If the result set has records, an email with the output attached is sent to the specified recipients. If it has no records, no email is sent.
- **Conditional E-Mail Output (Inline)**  
If the result set has records, an email is sent to the specified recipients with the output contained in the email body. If it has no records, no email is sent.
- **Conditional Publish**  
If the result set has records, the output will be available from the Jobs page. A notification email is sent to the specified recipients if this is configured. If the result set has no records, no output is generated and no email is sent.
- **Cache ResultSet (Append)**  
Cache the result set in a database table (Append)
- **Cache ResultSet (Delete&Insert)**  
Cache the result set in a database table (Delete&Insert)
- **Print**  
Prints the report output to the default printer
- **Burst**  
Generates report files for each distinct value in a resultset. This requires that the query for the report be ordered by the first column, which will be the burst-id column. When the value of this column changes, a new file is generated. Burst jobs are only possible with **Tabular** reports. The files generated will have the burst-id in their file names so different files can be identified and copied or ftped to different locations for example.

**Note:**

- If the output for a job creates a file e.g. publish to pdf, this file is stored in the **ART\_HOME\WEB-INF\work\export\jobs** directory

## 22.3. Archives

For publish and conditional publish jobs, you can specify that a certain number of job runs should be archived i.e. files generated from past runs should be available for viewing. This is done by setting the **Number of Runs to Archive** field when defining the job, and archived files are available from the **Archives** menu.

## 22.4. Job Auditing

Use the **Enable Auditing** option to enable/disable logging of additional information for a job run. If unchecked, only the last start and end time is recorded (in the ART\_JOBS table). If checked, an additional record will be created in the ART\_JOBS\_AUDIT table indicating the start time, end time and status of each job run.

## 22.5. Saved Schedules

If a schedule is used frequently, it can be saved such that you don't need to enter all the details each time you create a job. To reuse a schedule, select it from the **Schedules** option and click on the **Get** button. The schedule details will be retrieved and filled in the appropriate job schedule fields. Administrators can create, modify or delete schedules using the **Configure | Schedules** menu.

## 22.6. Job Duration

When defining the job schedule, you can specify the date when the job should start running and the date when it should stop in the **Start Date** and **End Date** fields respectively. If the start date is left blank, the job will start running on the current date, as per its schedule. If the end date is left blank, the job will continue to run indefinitely.

## 22.7. Shared Jobs

By default, only the owner of a job has access to its output, particularly for published jobs. Shared jobs allow the output of a job to be shared with other users. To share a job's output, check the **Allow Sharing** field and select the users or user groups with whom the output will be shared using the **Configure | Access Rights** menu. These users will be able to access the job's output from the Jobs menu.

### 22.7.1. Split Jobs

If the report for a shared job uses rules, you can specify that the rule values for the shared users be applied so that each individual user may get different, individualized output. To do this, check the **Allow Splitting** field. If this field is unchecked and the report uses rules, the rule values of the job owner will be applied and all shared users will get the same output. If the report doesn't use rules, the value of this field will have no effect. A single, identical output will be available to all users.

## 22.8. Random Start Times

You may not care when exactly a job runs, as long as it runs in a certain window of time. If you leave the Hour and Minute fields blank when creating the job, the job will be assigned a random hour between 3-6 and a random minute between 0-59 in which it will run. This may be useful for jobs that you require to run at night. Additionally, you can specify an explicit time range in which the job should run. To do this, in the **Hour** field, define the required start time and end time separated by | e.g. 4|7, 4:30|6, 12:45|13:15. The job will then be assigned a random execution time in this range.

## 23. Cached Results

ART can be used to reverse query results from a datasource to the same or a different datasource. This allows administrators to:

- Group data to summary tables, creating a data mart from which to develop other reports
- Create Data-Transformation-Load procedures (DTL) to feed other systems

One typical usage is when a large dataset would create performance issues if queried directly by end users. If the underlying data can be grouped - for instance on a monthly basis - a Cached Result job can summarize the information and then fast-performing queries can be created to analyse the summarized data.

### 23.1. Create a new Cached Result

In order to cache the results of an existing report:

- Schedule a job for the report
- In the Add Job screen among the **Job Type** items, the following items appear for administrators:
  - **Cache ResultSet (Append)**
  - **Cache ResultSet (Delete & Insert)**
- Select the datasource where to reverse the data (note: the datasource must be writeable i.e. the datasource user should have UPDATE/DELETE/INSERT/CREATE TABLE rights on the target database table)
- In the **Cached Table Name** field, enter the name of the table in the target datasource that will contain the data
- Schedule the timing when the action should take place, like any other job

### 23.2. Accessing a Cached Result

At the given time ART will:

- Run the query
- Connect to the datasource where to reverse the results
  - The Cached Table is created in the selected datasource (if the table doesn't exist). The column names will match with the query column names. Blanks or other special characters in the names are replaced with "\_".
  - The table content is deleted if the option **Cache ResultSet (Delete & Insert)** was selected
- Reverse all the query results in the new table

A cached table is just a normal table in the database. Details about the columns names are available in the Jobs page.

#### Note:

- The Cached Table is removed when the job is deleted. However if the job is edited and the table name changed, the previous table is maintained.
- No indexes are created on top of the Cached Table. For large datasets to be queried, it is suggested to manually create indexes once the table is created. Since the table is not dropped/recreated at each Job execution the indexes will persist.

- Basic columns types (integer, varchar, date, etc) can be properly reversed. BLOB, CLOB or other database specific non-standard data types have not been tested and will likely generate errors.

## 24. Dynamic Recipients

Dynamic recipients allows you to email report results to a dynamic list of people. This may be useful where the recipients may change from day to day or week to week e.g. sending an email to staff who are low on their targets. It may also be useful to send filtered report results e.g. send to managers only the sales for their branch.

Using the dynamic recipients feature will involve the following process

- Create the main report with the data to be sent to the desired recipients
- Create a report of type **Dynamic Job Recipients** that will contain details of the recipients, including their email addresses
- Schedule a job for the main report and select the dynamic recipients report you created in the **Dynamic Recipients** field

There are several ways to use dynamic recipients.

### 24.1. Dynamic Recipients only

If you want all the recipients to get the same data and the same email message, then define a dynamic recipients report with only one column. This column should contain the email addresses of the recipients. The name of the column doesn't matter. Example:

```
SELECT email
FROM employee
```

### 24.2. Dynamic Recipients + Personalization

If you want all the recipients to get the same data, but you would like some personalization of the email message e.g. having a greeting like Dear John instead of Dear Employee, then define a dynamic recipients report where the first column contains the recipient email addresses, and any additional columns that you would like to use in the email message e.g.

```
SELECT email, first_name, other_name last_name, order_count
FROM employee, orders
WHERE employee.id=orders.employee_id
```

When defining the job for your data query, in the email message section, you can then use column labels as placeholders where personalized details will be put. The labels consist of column names from the recipients report surrounded by # signs e.g. you can have a message like

```
Dear #first_name# #last_name#
You are now at #order_count# orders, which is below the target of 100.
```

Each person in the dynamic recipients report list will get a personalized email with the column labels substituted with his values.



## 24.3. Dynamic Recipients + Filtering

If you want the recipients to get different data, you will define a dynamic recipients report where the first column contains the recipient email addresses, any additional columns with personalization information if you want, and 2 other special, hard coded columns named **recipient\_column** and **recipient\_id**. These will be used to filter the data query. i.e. WHERE <recipient\_column> = <recipient\_id>. The main, data query will also need to have a special, hard coded label, **#recipient#**, in the where clause.

Example:

If we want to email users only transactions that they created, our data query can be something like

```
SELECT *
FROM transactions
WHERE transaction_date = CURDATE()
AND #recipient#
```

We can then define a dynamic recipients report like

```
SELECT email, "transaction_user" recipient_column, employee_id recipient_id
FROM employee
```

This will cause the **#recipient#** placeholder in the data query to be replaced with the condition `transaction_user = <employee_id>`, where `<employee_id>` will be different for each recipient, as per the dynamic recipients query. We can then schedule a job for the main report and set the dynamic recipients field, and all the recipients will get a separate email with their specific data only.

If we also want to include personalization fields in the email message body, we can add these to the dynamic recipients query e.g

```
SELECT email, "transaction_user" recipient_column, employee_id recipient_id, first_name
FROM employee
```

and we can then include the personalization placeholders in the email message field as desired.

**Note:**

- With the personalization and filtering options, you can include columns with the name **email\_cc** or **email\_bcc** to specify email addresses that should be included in the cc or bcc fields of the email sent.
- Email address columns can have multiple emails separated by ;

## 25. FTP Servers

You can configure ftp servers to be used after a publish job is generated, to ftp the generated file to a given ftp server.

Use the **Configure | FTP Servers** menu to manage ftp server configurations. When Adding an ftp server, the following fields are available.

Field	Comment
<b>ID</b>	A unique ID for the ftp server configuration, generated by ART
<b>Name</b>	A name for the ftp server configuration. This will be used in the Add Job page in the list of available ftp servers.
<b>Description</b>	A description for the ftp server configuration
<b>Active</b>	Whether this configuration is active or not. If it is not active, no files will be ftped for jobs that use this ftp server configuration.
<b>Server</b>	The IP address or host name of the ftp server
<b>Port</b>	The ftp port of the server
<b>User</b>	The ftp user to use to connect to the server
<b>Password</b>	The ftp user's password
<b>Remote Directory</b>	If the files are to be ftped to the user's home directory, then this field should be left blank. If the files are to be ftped to another directory, then the path should be given here, always beginning with "/" e.g. /sub-folder-a or /sub-folder-a/sub-sub-folder-b.

## 26. Integrated Windows Authentication

ART uses the spnego library available at <http://spnego.sourceforge.net/> to provide Integrated Windows Authentication functionality. These instructions are largely based on the documentation found on that project's website.

### 26.1. Prerequisites

You need to have at least 3 separate machines as follows

- Active Directory server (or other KDC server)
- Application server (a different machine where the application server e.g. Tomcat is installed)
- Client machine(s) (from where you'll access ART)

### 26.2. On the Active Directory server

- Create a user in AD to be used for authentication purposes. This user doesn't need to have access to log in to computers. e.g. a user named spnego. Set the password to never expire
- Create spns that point to the application server/spnego user combination using the setspn command. Use syntax like below

```
setspn -A HTTP/app-server spnego
setspn -A HTTP/app-server.domainname spnego
```

Replace **app-server** with the appropriate machine name of the application server, **my.domain.com** with the domain name and **spnego** with the username of the domain account to be used for spnego access. If you'll also be accessing the web application on the application server via ip address e.g. `http://192.168.56.101:8080/art`, also create spns for the ip address. Examples

```
setspn -A HTTP/app-server spnego
setspn -A HTTP/app-server.my.domain.com spnego
setspn -A HTTP/192.168.56.101 spnego
setspn -A HTTP/192.168.56.101.my.domain.com spnego
```

### 26.3. On the Application server

- Create a file in the ART\_HOME\WEB-INF directory named **login.conf** with the following details

```
spnego-client {
    com.sun.security.auth.module.Krb5LoginModule required;
};

spnego-server {
    com.sun.security.auth.module.Krb5LoginModule required
    storeKey=true
    isInitiator=false;
};
```

- Create a file in the ART\_HOME\WEB-INF directory named **krb5.conf** with the following details. Replace **MY.DOMAIN.COM** with your domain name. For the kdc parameter, use the fully qualified domain name of the AD server.

```
[libdefaults]
    default_realm = MY.DOMAIN.COM
    default_tkt_enctypes = aes128-cts rc4-hmac des3-cbc-sha1 des-cbc-md5 des-cbc-crc
    default_tgs_enctypes = aes128-cts rc4-hmac des3-cbc-sha1 des-cbc-md5 des-cbc-crc
    permitted_enctypes = aes128-cts rc4-hmac des3-cbc-sha1 des-cbc-md5 des-cbc-crc

[realms]
    MY.DOMAIN.COM = {
        kdc = ad-server.my.domain.com
        default_domain = MY.DOMAIN.COM
    }

[domain_realm]
    .MY.DOMAIN.COM = MY.DOMAIN.COM
```

- Edit the ART\_HOME\WEB-INF\web.xml file and add a filter as below. Replace the **spnego.preauth.username** and **spnego.preauth.password** parameter values with the details of the domain account created to enable spnego access. Replace the **spnego.krb5.conf** and **spnego.login.conf** parameter values with the full path of the respective files. Leave all the other parameters as they are. The spnego filter mapping must come before other filter mapping elements.

```
<filter>
    <filter-name>SpnegoHttpFilter</filter-name>
    <filter-class>net.sourceforge.spnego.SpnegoHttpFilter</filter-class>

    <init-param>
        <param-name>spnego.allow.basic</param-name>
        <param-value>true</param-value>
    </init-param>

    <init-param>
        <param-name>spnego.allow.localhost</param-name>
        <param-value>false</param-value>
    </init-param>

    <init-param>
        <param-name>spnego.allow.unsecure.basic</param-name>
        <param-value>true</param-value>
    </init-param>

    <init-param>
        <param-name>spnego.login.client.module</param-name>
        <param-value>spnego-client</param-value>
    </init-param>

    <init-param>
        <param-name>spnego.krb5.conf</param-name>
        <param-value>C:\tomcat\webapps\art\WEB-INF\krb5.conf</param-value>
    </init-param>

    <init-param>
        <param-name>spnego.login.conf</param-name>
        <param-value>C:\tomcat\webapps\art\WEB-INF\login.conf</param-value>
    </init-param>

    <init-param>
        <param-name>spnego.preauth.username</param-name>
        <param-value>spnego</param-value>
    </init-param>
```

```

<init-param>
  <param-name>spnego.preauth.password</param-name>
  <param-value>spnego</param-value>
</init-param>

<init-param>
  <param-name>spnego.login.server.module</param-name>
  <param-value>spnego-server</param-value>
</init-param>

<init-param>
  <param-name>spnego.prompt.ntlm</param-name>
  <param-value>true</param-value>
</init-param>

<init-param>
  <param-name>spnego.logger.level</param-name>
  <param-value>1</param-value>
</init-param>
</filter>
<filter-mapping>
  <filter-name>SpnegoHttpFilter</filter-name>
  <url-pattern>*.jsp</url-pattern>
</filter-mapping>

```

## 26.4. On the client machine

- The **Default Authentication Method** for ART needs to have been set to **Auto** (done in the Settings page)
- Login to a client machine using a domain account
- Access the ART home page as usual

### 26.4.1. *Omitting the credentials box*

- **Firefox**  
By default, firefox will still display a credentials box requiring a user to enter their domain user-name and password. To avoid this, do the following
  - In the address bar, type **about:config**
  - In the filter box, type **network.negotiate**
  - Double click on the **network.negotiate-auth.trusted-uris** option and enter the url of the application server (excluding the http part and including port number if not port 80) e.g. app-server:8080

If the credentials box is still displayed, set the following options in a similar way

- **network.negotiate-auth.delegation-uris**
- **network.automatic-ntlm-auth.trusted-uris**
- **IE**  
For IE, the credentials box may be displayed if you access the web application using an ip address. To avoid this, do the following

- Under internet options, security, local intranet, sites, click on advanced and add the url to the application server e.g. `http://192.168.56.101:8080`

## 26.5. Using a keytab file

Instead of having the spnego username and password in plain text in the web.xml file, you can use a keytab file to hold these credentials. Take the following steps on the application server

- Stop tomcat
- Copy the file `ART_HOME\WEB-INF\krb5.conf` to the windows directory e.g. `C:\windows`
- Rename the file to `krb5.ini`
- Open a command prompt window, cd to the `ART_HOME\WEB-INF` directory and type a command with syntax like the following

```
ktab -a <spnego user> <spnego password> -k <file name>
```

E.g.

```
ktab -a spnego spnego -k art.keytab
```

- Edit the `ART_HOME\WEB-INF\login.conf` file to have contents like the following. Set the full path to the keytab file and the spnego username (principal) as per your configuration

```
spnego-client {
    com.sun.security.auth.module.Krb5LoginModule required;
};

spnego-server {
    com.sun.security.auth.module.Krb5LoginModule required
    storeKey=true
    useKeyTab=true
    keyTab="file:///c:/tomcat/webapps/art/WEB-INF/art.keytab"
    principal=spnego;
};
```

- Edit the `ART_HOME\WEB-INF\web.xml` file. Make the **spnego.preauth.username** and **spnego.preauth.password** parameters blank. i.e.

```
<init-param>
    <param-name>spnego.preauth.username</param-name>
    <param-value></param-value>
</init-param>

<init-param>
    <param-name>spnego.preauth.password</param-name>
    <param-value></param-value>
</init-param>
```

- Restart tomcat

You should have integrated authentication as before

## 26.6. Changing the spnego user

If you need to change the AD user used to enable spnego access, first delete the spns associated with the application server and then create new ones for the desired user. An spn for a given server can only refer to a single user. To delete the spns, you can use syntax similar to the following

```
setspn -D HTTP/app-server spnego  
setspn -D HTTP/app-server.my.domain.com spnego
```

## 27. CAS Authentication

To use CAS authentication, edit the **web.xml** file and uncomment the items in the CAS configuration section, providing appropriate urls for the CAS server and for ART.

Also, from the Settings page, set the **Default Authentication Method** for ART to **CAS** and set the **CAS Logout URL** to enable users to log out of CAS after logging out of ART.



## 28. Application Logs

ART uses the [Logback](#) library for application logging. By default, application logging is done to standard output and includes logging on errors, warnings and information messages. The location of the logs or the amount of logging generated can be modified by making appropriate configuration changes to the ART\_HOME\WEB-INF\classes\logback.xml file. Changes made to the logging configuration e.g. changing certain logging levels from "info" to "debug" automatically take effect after the configured **scanPeriod** (values can be specified as milliseconds, seconds, minutes or hours. See <http://logback.qos.ch/manual/configuration.html> for more details).

In addition to being available on the application server's standard output log files, e.g. stdout.log or catalina.log on Tomcat, application logs can also be viewed from within ART. This means that you don't need to have access to the application server machine in order to view application logs.

Application logs can be viewed from the **Logs** menu. New logs are added to the bottom and you'll need to refresh the page to view them. Also note that this page doesn't display all the logs ever generated by the application, only the most recent ones.

### 28.1. SQL Logging

You can set up logging for the sql that is generated when queries are run, e.g. to see the final sql generated, or to see how long queries take to execute. ART uses the [log4jdbc-log4j2](#) library to enable such logging. Take the following steps to configure sql logging.

- Select the **Configure | Datasources** menu and **Edit** the datasource you're interested in
- Select **SQL Logging** in the Database Type field. This will set the JDBC Driver and JDBC fields appropriately for sql logging. Alternatively, you can set the values manually as follows. Set the JDBC Driver field to `net.sf.log4jdbc.sql.jdbcapi.DriverSpy` and modify the JDBC URL field by prepending **jdbc:log4** to the existing url, e.g. resulting in a url like `jdbc:log4jdbc:mysql://localhost/mydb`
- **Save** the datasource details
- Modify the ART\_HOME\WEB-INF\classes\logback.xml file and add loggers for the items you are interested in e.g.

```
<logger name="jdbc.sqltiming">
  <level value="info" />
</logger>
```

- The available loggers and what events they log can be found on the log4jdbc-log4j2 project home page, <http://log4jdbc.brunorozendo.com/>. ART uses the SLF4J configuration.
- That's all. The logging information will now be included in the application logs when reports are run against that datasource.

## 29. Tomcat Configuration

### 29.1. Memory options

If you are using Tomcat as the application server, there are some configuration items you can set to improve performance. This is mainly setting Tomcat to run in server mode and increasing the amount of memory available to Tomcat e.g. so that jobs don't run out of memory.

#### 29.1.1. Windows

If Tomcat was installed as a service

- Run the TOMCAT\_HOME\bin\tomcat8w.exe (or similar file for your Tomcat version). This may need to be run as administrator.
- In the Java tab, in the **Java Virtual Machine** section, set Tomcat to run in server mode by changing the jvm.dll to the one in the JDK e.g. C:\Program Files\Java\jre1.8.0\_25\bin\server\jvm.dll
- Increase the amount of memory available to Tomcat by setting a value in the **Maximum memory pool** textbox e.g. 1024 (this value shouldn't be very large compared to the total amount of memory available on the machine, otherwise the operating system and other applications may be starved of RAM)

**Note:**

- The "service user account" configured to start and stop the Tomcat service needs to have appropriate permissions, otherwise you may get errors when accessing ART. In particular, ART requires write access to the ART\_HOME\WEB-INF\work and ART\_HOME\js-templates directories.

If Tomcat is run from a batch file

- Create a file named **setenv.bat** in the TOMCAT\_HOME\bin directory (or edit it if it exists) and set the configuration options in the JAVA\_OPTS environment variable e.g.

```
set JAVA_OPTS=-server -Xmx1024m
```

#### 29.1.2. Linux

Create a file named **setenv.sh** in the TOMCAT\_HOME/bin directory (or edit it if it exists) and set the configuration options in the JAVA\_OPTS environment variable e.g.

```
export JAVA_OPTS="-server -Xmx512m"
```

### 29.2. Accessing Tomcat via Apache

You can use the Apache HTTP Server as a front end to your application that resides on Tomcat. This can be done for a number of reasons, including clustering or enabling access to the application from a more familiar url.

There are a number of ways to run Tomcat behind Apache.

### 29.2.1. Using *mod\_proxy*

- Ensure the following lines in your Apache **httpd.conf** file are uncommented

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

- Add an item at the end of the **httpd.conf** file to indicate how ART will be accessed by users and how Apache will communicate with Tomcat e.g

```
ProxyPass /art http://localhost:8080/art
ProxyPassReverse /art http://localhost:8080/art
```

That's it. Start Apache and Tomcat (the order of starting and stopping doesn't matter), and now you can access ART from the url localhost/art (i.e <apache-server>/art).

### 29.2.2. Using *mod\_proxy\_ajp*

If you have Apache 2.2 and above, you can use mod\_proxy\_ajp

- Ensure the following lines in your Apache **httpd.conf** file are uncommented

```
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

- Ensure you have an AJP connector defined in the **TOMCAT\_HOME\conf\server.xml** file e.g.

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

- Add an item at the end of the **httpd.conf** file to indicate how ART will be accessed by users and how Apache will communicate with Tomcat e.g

```
ProxyPass /art ajp://localhost:8009/art
ProxyPassReverse /art ajp://localhost:8009/art
```

- Note that this definition uses the ajp protocol and port 8009, which is the port defined for the AJP connector in Tomcat's server.xml

That's it. Start Apache and Tomcat (the order of starting and stopping doesn't matter), and now you can access ART from the url localhost/art (i.e <apache-server>/art).

### 29.2.3. Using *mod\_jk*

If you need more powerful proxying features, you can download, configure and use the mod\_jk connector. This is slightly more involving than the other two methods already mentioned and you can use the instructions available at [http://www3.ntu.edu.sg/home/ehchua/programming/howto/ApachePlusTomcat\\_HowTo.html](http://www3.ntu.edu.sg/home/ehchua/programming/howto/ApachePlusTomcat_HowTo.html) to get it working.

**Note:**

- For Ubuntu/Debian, instead of using LoadModule to enable the modules, use a2enmod from the command line, e.g.

```
a2enmod proxy_ajp  
a2enmod proxy  
a2enmod proxy_http
```

- Also for Ubuntu/Debian, add the ProxyPass and ProxyPassReverse items to the **apache2.conf** file instead of httpd.conf.

## 30. Customizing ART

You can customize ART to fit your needs.

### 30.1. Using live files

You can customize or update ART files directly on the application server where ART is deployed. This may be especially useful for minor changes.

#### 30.1.1. Setting up Apache Ant

If you are going to customize the application source code, you may want to setup Apache Ant to make compiling your changes easier. To do this, take the following steps

- Download the Apache Ant zip package from <http://ant.apache.org/>
- Extract the zip file to C:\, or other directory of your choice. A new directory will be created e.g. C:\apache-ant-1.8.1
- Add the Ant bin directory to the PATH environment variable e.g. C:\apache-ant-1.8.1\bin

#### 30.1.2. Customizing java files

Java source files are contained in sub directories under the **ART\_HOME\WEB-INF\classes\art** directory. To make customizations to these files, take the following steps.

- Configure Apache Ant as described above
- Make changes to the desired .java files
- Open a command prompt window and navigate to the **TOMCAT\_HOME\webapps\art** directory
- Type the command `ant compile`
- The modified .java files will be recompiled
- Reload the ART application using the Tomcat Application Manager, or restart tomcat
- The application will now use the modified source code

#### 30.1.3. Customizing jsp files

JSP files can be modified and results immediately viewable when the relevant page is refreshed. If you don't see the expected changes when you refresh the page, the page displayed may be a cached copy. You may need to clear the application server cache and the browser cache to ensure the page is refreshed. If using Tomcat, you can clear the tomcat cache by deleting the **TOMCAT\_HOME\work\Catalina\localhost\art** directory. If using Firefox, you can clear the browser cache from the History | Clear Recent History menu.

#### 30.1.4. Customizing other files

Other files that affect how ART works can also be modified. Examples include

- ART\_HOME\WEB-INF\classes\quartz.properties - Change quartz configuration items e.g. number of threads
- ART\_HOME\WEB-INF\classes\logback.xml - Change logging level for quartz or ART classes

### 30.1.5. Updating jasperreports

If you are using a version of iReport that is different from the jasperreports version contained in ART, you may want to update the jasperreports version contained in ART to match your iReport version. This may not be necessary, but would be useful if you encounter problems while running reports. Take the following steps to update the jasperreports library and template files.

- Download the required jasperreports .jar from the jasperreports sourceforge page, <http://sourceforge.net/projects/jasperreports/files/jasperreports/> e.g download the file jasperreports-5.0.1.jar from the JasperReports 5.0.1 folder.
- Ensure Apache Ant is set up as described above
- Stop Tomcat
- Backup the contents of the ART\_HOME\WEB-INF\work\templates directory, where the jasperreport template files reside
- Delete the jasperreports .jar file contained in the ART\_HOME\WEB-INF\lib directory
- Copy the just downloaded jasperreports .jar file to the ART\_HOME\WEB-INF\lib directory
- Open a command prompt window and navigate to the ART\_HOME directory
- Type the command `ant clean-jasperreports`
- Type the command `ant compile-jasperreports`
- Restart Tomcat

Your reports should now work properly.

## 30.2. Using Ant

If you don't wish to modify the live files, e.g. you want to test the changes on a test server first, you can modify the files in another location and create an application .war file to deploy on your test environment.

### 30.2.1. Using Ant without an IDE

- Ensure Apache Ant is set up as described above
- Unzip the PACKAGE\_PATH\art.war e.g. unzip to a new directory PACKAGE\_PATH\art
- Using a text editor, make modifications to the required files
- Open a command prompt window and navigate to the PACKAGE\_PATH\art directory
- Type the command `ant war`
- A new file named art.war will be created in the PACKAGE\_PATH\art directory. Deploy this file to your desired application server.

### 30.2.2. Using Ant with NetBeans

Instead of using a text editor to make changes and Ant to compile and package the changes, you can use an IDE to do this. The following are sample steps using NetBeans 7.2.1.

- Unzip the PACKAGE\_PATH\art.war file e.g. unzip to a new directory PACKAGE\_PATH\art. You can unzip the war file the same way as any zip file e.g. using 7zip.
- Create a directory to hold NetBeans project files e.g. C:\MyNetBeansProjects\art
- Open NetBeans
- Select the **File | New Project** menu
- Select the **Java Web** category and **Web Application with Existing Sources** project and click on the **Next** button
- For the **Location** field, select the PACKAGE\_PATH\art directory. For the Project Folder field, select the directory you created for this e.g. C:\MyNetBeansProjects\art. Click on the **Next** button.
- Select the Server to use e.g. Apache Tomcat and Java EE version e.g. Java EE 7 Web. Click on the **Next** button.
- The Existing Sources and Libraries dialog should be pre-populated correctly with the appropriate paths. Click on the **Finish** button. If prompted, choose to delete existing .class files.
- Make changes to the files as appropriate
- To test the changes, use the Run Project icon on the menu bar, or right click on the project in the Projects explorer and select Run.
- To debug the code, set breakpoints as appropriate by clicking on the edge of the editor at the desired line of code and use the Debug Project icon on the menu bar or right click on the project and select Debug. If the debug process doesn't start with NetBeans indicating that it is waiting for tomcat to stop, open Task Manager and kill the java.exe process.
- To generate a war file to deploy, right click on the project and select Clean and Build. The generated art.war file will be located in the C:\MyNetBeansProjects\art\dist directory. Deploy this file to your desired application server

## 30.3. Using Maven

You can take the following steps to modify ART source code, using Apache Maven as the build tool. ART source files with a maven structure will be found in the PACKAGE\_PATH\src directory.

- Install Maven
- (Optional) Install a maven repository manager e.g. [Sonatype Nexus](#). A repository manager is optional but greatly increases productivity.

### 30.3.1. Using Maven without an IDE

- Using a text editor, make modifications to the required files
- Open a command prompt window and navigate to the PACKAGE\_PATH\src\art-parent directory
- Type the command `mvn clean package`
- A new file named art.war will be created in the PACKAGE\_PATH\src\art-parent\art\target directory. Deploy this file to your desired application server.

### 30.3.2. Using Maven with NetBeans

The following are sample steps using NetBeans 7.2.1.

- Open NetBeans
- Select the **File | Open Project** menu
- Select the `PACKAGE_PATH\src\art-parent` directory in the left hand panel, and click on the **Open Required Projects** box on the right hand panel. Click on the **Open Project** button.
- Make changes to the files as appropriate
- To test the changes, right click on the art project in the Projects explorer and select Run. Select the server on which to deploy the application and click on OK.
- To generate a war file to deploy, right click on the Parent project and select Clean and Build. The generated art.war file will be located in the `PACKAGE_PATH\src\art-parent\art\target` directory. Deploy this file to your desired application server

### 30.3.3. Using Maven with Eclipse

The following are sample steps using Eclipse Juno (4.2) SR2.

- Download the Eclipse IDE for Java EE developers package
- Unzip the package to your desired location e.g. C:\, to generate a C:\eclipse directory. Run the C:\eclipse\eclipse.exe file.
- Ensure you have an internet connection
- Select the **Help | Eclipse Marketplace** menu
- On the Search tab, in the Find field, type maven and hit enter
- Find the "Maven Integration for Eclipse" plugin (m2e by eclipse.org) and click on the Install button
- Once the plugin installs, go back to the Eclipse Marketplace and again type maven in the Find field and hit enter
- Find the "Maven Integration for Eclipse WTP" plugin (m2e-wtp by eclipse.org) and click on the Install button.
- Create a directory to hold eclipse workspace files e.g. C:\MyEclipseWorkspaces\art
- In Eclipse, select the **File | Switch Workspace | Other** menu
- Select the workspace folder you created e.g. C:\MyEclipseWorkspaces\art and click on OK. Wait for Eclipse to restart
- Select the **File | Import** menu
- Open the Maven group, select the **Existing Maven Projects** option and click on Next.
- Set the **Root Directory** field to the `PACKAGE_PATH\src\art-parent` directory. This should list the art-parent project, with the artdbcp, artmail and art projects beneath it. If the projects aren't retrieved, try to hit the enter key after typing the path in the root directory field. Ensure all the projects are selected and click on Next.
- Click on Finish in the final screen.
- Click on the Restore icon in the left most panel to have the project windows displayed on the screen and close the welcome page.
- Make changes to the files as appropriate
- Ensure you have a JDK installed
- Select the **Window | Preferences** menu. Expand the **Java** group and select the **Installed JREs**



option. In the right hand panel, select the default JRE and click on the Edit button. Set the **JRE home** field to a JDK folder e.g. C:\Program Files\Java\jdk1.8.0\_20. Instead of editing the default JRE you can also use the Add button to add a new one and then check it as the default workspace JRE.

- Right click on the art-parent project and select the **Run As | Run Configurations** menu. Select the **Maven Build** option, and then click on the **New launch configuration** icon at the top of the list. Give a **Name** to the configuration e.g. package and set the **Base directory** to the PACKAGE\_PATH\src\art-parent directory. In the **Goals** field, type clean package. In the JRE tab, ensure the JRE selected is the one defined earlier. Click on the Apply button. To generate a war file immediately, click on the Run button.
- To test changes, set up a server. See <https://github.com/OneBusAway/onebus-away/wiki/Setting-Up-a-Tomcat-Server-in-Eclipse> . Convert the art webapp project to a Dynamic Web Module and run the application. See <https://wiki.base22.com/display/btg/How+to+create+a+Maven+web+app+and+deploy+to+Tomcat++fast>
- To generate a war file to deploy, click on the drop down on the **Run As** icon in the menu bar, and select the configuration you created e.g. package. The generated art.war file will be located in the PACKAGE\_PATH\src\art-parent\art\target directory. Deploy this file to your desired application server

## 30.4. Custom Export Path

By default, if the output for a report creates a file e.g. pdf, this file is stored in the ART\_HOME\WEB-INF\work\export directory. You can designate a different directory to be used as the export path e.g. to a location on the application server that contains more disk space. To do this, modify the ART\_HOME\WEB-INF\art-custom-settings.json file and set the path in the **exportDirectory** field.

## 30.5. Translating ART

You can translate ART so that the user interface is displayed in your own language. Follow the following steps to translate ART.

- Download the latest ART package, unzip it and unzip the **art.war** file using any zip/unzip software. Get the **ArtMessages.properties** file from the WEB-INF\i18n directory and make a copy of it, naming your new file in the format **ArtMessages\_xx.properties**, where xx is the ISO 639-1 language code for your language. A List of these language codes can be found here - <http://www.mathguide.de/info/tools/languagecode.html>. An example would be ArtMessages\_pt.properties for Portuguese. If your language is written differently in different countries you can add the 2 letter country specifier to the file name so that you have a file name like ArtMessages\_pt\_BR.properties for Brazilian Portuguese. You can find a list of country codes here - [http://www.immigration-usa.com/country\\_digraphs.html](http://www.immigration-usa.com/country_digraphs.html).
- Change all the text in your new properties file after the = signs to your language
- If your language uses a non ISO-8859 character set (e.g. chinese, arabic, russian etc) name the file ArtMessages\_xx-UTF8.properties and then decode it to an ISO-8859 character set with UTF-8 escapes using the **native2ascii** tool that is available with your Java installation. An example command is as below.

```
native2ascii ArtMessages_xx-UTF8.properties ArtMessages_xx.properties
```

- Once you have your translated properties file in the i18n folder, you can restart the application server to make the new translation available to the application, or wait for 1 hour after which the messages are refreshed.
- Now you can log in to the application and include the **lang=xx** parameter in browser url to activate the new language e.g. `http://localhost:8080/art/reports?lang=xx`
- If you would like the new language to be listed for selection within the application e.g. on the login page, add the language code and the language name to the **ART\_HOME\WEB-INF\languages.properties** file, save the file and restart the application server.
- If in addition you would like this new translation to be included in future ART versions, email the properties file to **tanyona** (at) **users.sf.net** or create a new post on the [Discussion forum](#) and attach the file there.

## 31. Support

- If you have any questions or comments, post these on the ART Help Discussion forum, <http://sourceforge.net/p/art/discussion/352129/>