

ART Manual

3.0-alpha1

Table of Contents

1. Introduction	2
1.1. Overview	2
1.2. Administration Overview	2
2. Settings	4
3. Datasources	7
3.1. Creating a new datasource	7
3.2. Some JDBC Drivers and URLs	8
4. Users	11
4.1. Setting up a user	11
4.2. Setting up the Public User	11
5. Authentication	12
5.1. Internal Authentication	12
5.2. External Authentication	12
5.2.1. Windows Authentication	12
5.2.2. LDAP Authentication	13
5.2.3. Database Authentication	13
5.2.4. Custom Authentication Sources	14
6. User Groups	15
7. User Group Membership	16
8. Query Groups	17
9. Queries	18
9.1. Introduction	18
9.2. Creating a query	18
9.3. Query Types	20
9.4. Parameters	26
9.5. Executing a query via URL	28
9.6. View Modes	30
9.7. Some Useful Queries	31
10. Multiple Statements	32
11. LOV Queries	33
11.1. Example	33
12. Dynamic Queries	35
12.1. Operators	35
12.2. Dynamic Query Examples	36
13. Drill Down Queries	38
13.1. Defining the main query	38
13.2. Defining the drill down query	38
13.3. Example	38
13.4. Using the main query's parameters	40
14. Dynamic Query Datasources	42
15. RSS Feeds	43
16. Dashboards	45
16.1. Features	46
16.2. Syntax	46
17. Text Queries	48
18. jXLS Spreadsheets	49
19. Pivot Tables (OLAP)	50
19.1. Mondrian	50

19.2. Mondrian XMLA	50
19.3. Microsoft XMLA	50
20. User Privileges	52
21. Admin Privileges	53
22. Rules	54
23. Chained Parameters	56
23.1. Example	56
23.2. Default Values	59
24. Scheduling	60
24.1. Scheduling a new job	60
24.2. Job Types	61
24.3. Job Archives	62
24.4. Job Auditing	62
24.5. Saved Schedules	62
24.6. Job Duration	63
24.7. Shared Jobs	63
24.7.1. Split Jobs	63
24.8. Random Start Times	63
25. Cached Results	64
25.1. Create a new Cached Result	64
25.2. Accessing a Cached Result	64
26. Dynamic Recipients	66
26.1. Dynamic Recipients only	66
26.2. Dynamic Recipients + Personalization	66
26.3. Dynamic Recipients + Filtering	67
27. Single Sign On	68
27.1. Prerequisites	68
27.2. On the Active Directory server	68
27.3. On the Application server	68
27.4. On the client machine	70
27.4.1. Omitting the credentials box	70
27.5. Using a keytab file	71
27.6. Changing the spnego user	72
28. Application Logs	73
28.1. SQL Logging	73
29. Tomcat Configuration	74
29.1. Memory options	74
29.1.1. Windows	74
29.1.2. Linux	74
29.2. Accessing Tomcat via Apache	75
29.2.1. Using mod_proxy	75
29.2.2. Using mod_proxy_ajp	75
29.2.3. Using mod_jk	76
30. Customizing ART	77
30.1. Using live files	77
30.1.1. Setting up Apache Ant	77
30.1.2. Customizing java files	77
30.1.3. Customizing jsp files	77
30.1.4. Customizing other files	78
30.1.5. Updating jasperreports	78

30.2. Using Ant	78
30.2.1. Using Ant without an IDE	78
30.2.2. Using Ant with NetBeans	79
30.3. Using Maven	79
30.3.1. Using Maven without an IDE	79
30.3.2. Using Maven with NetBeans	80
30.3.3. Using Maven with Eclipse	80
30.3.4. Compiling under Java 7	81
30.4. Custom Export Path	81
31. Support	83
31.1. Resources	83

ART Manual

1. Introduction

1.1. Overview

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

- **ART Administrators** define datasources, queries, users, privileges etc.
- **ART Users** execute queries and view the results in a browser or save the data in a variety of formats.


ART is open source software distributed under the [GPL license](#). You can install and use it for personal or commercial purposes without any charge.

1.2. Administration Overview

The Administrators define a number of items including

- **Settings** used to configure how ART works
- **Datasources** against which to execute queries
- **Query Groups** used to associate queries
- **Queries** (Tablular, Chart, Dashboard etc)
- **User Groups** used to manage users
- **Users**
- **User Group Membership** to add or remove users from user groups
- **User/User Group Privileges** to determine which users or user groups can view which queries
- **Admin Privileges** to determine which query groups and datasources administrators can manage
- **Rules** to be applied on queries if any
- **Rule Values** to be applied for specific users or user groups
- **Shared Jobs** to define which users can see which job outputs
- **Schedules** to manage schedules that can be used when creating jobs

These are defined and managed from the **Admin Console**.

Admin Console	
	
Settings	Define ART settings ART Repository, SMTP settings etc
Datasources	Define Datasource connection parameters define datasources and refresh or review connections status
Query Groups	Define Groups to which queries belong
Queries	Manage Queries create, update, delete, copy queries
User Groups	Define Groups to which users belong
Users	Define Users
User Group Membership	Assign users to user groups users can belong to zero, one, or many user groups
User/User Group Privileges	Grant/Revoke to users/user groups the right to access queries/query groups
Admin Privileges	Grant/Revoke to junior/mid Admins the right to act on query groups and datasources
Rules	Define Rule names
Rule Values	Set rule values for users/user groups
Shared Jobs	Grant/Revoke access to shared jobs
Schedules	Define schedules that can be used when creating jobs

The typical flow when defining a new query is

- Obtain the plain SQL statement and identify any parameters
- If the target datasource is not already defined, define it
- If new queries are needed in order to define lists of values (LOVs) to dynamically generate parameter values, define them
- Define the query and query parameters
- Modify user privileges to allow users to execute the query

2. Settings

Certain settings are used to configure how ART works. From the **Admin Console**, use the **Settings** option to manage ART settings.

Setting	Description
ART Database Username	Database username for the ART repository. The ART repository is the database used by ART to hold details like users, queries etc. The user needs SELECT, INSERT, UPDATE and DELETE rights on the ART tables.
ART Database Password	Database password for the ART repository.
ART Database JDBC Driver	JDBC driver name for the ART repository. If you are using a JNDI datasource, this must be blank.
ART Database JDBC URL	Database URL for the ART repository. 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>
Connection Pool Timeout (mins)	How long an idle connection should be maintained in the connection pool before being closed
Connection Test SQL	Short SQL query used to determine if a connection is alive e.g. "Select 1"
ART Administrator Email	Email address which is displayed in mailto link at the bottom of ART web pages
SMTP Server	Host name for the email server used to send emails
ART CSS (skin)	Path to CSS file used to determine the look of ART web pages. This file can be customized as desired.
Page Footer Logo	Image displayed in the footer of ART web pages
SMTP Username	Username to be used when sending emails if the email server is configured to require SMTP authentication. If SMTP authentication is not required by the email server, leave this blank.
SMTP Password	Password to be used when sending emails if the email server is configured to require SMTP authentication. If SMTP authentication is not required by the email server, leave this blank.
Use Secure SMTP	Defines whether to use secure SMTP when sending emails
SMTP Port	Port on which SMTP server is listening

Setting	Description
Show standard header and footer in public_user sessions	Show standard header (navigation links, login time etc) and footer for queries executed as public_user
PDF Document Page Size	Size and layout of documents generated by pdf output view mode
PDF Font Name	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 [wiki:Tips] documentation for details on how to use custom fonts with jasper reports.
PDF Font File	Path to a font file that contains the custom font
PDF Font Directory	Path to a directory that contains font files, one of which may be used in the pdf font name field
PDF Font Encoding	Encoding to use for the custom font
PDF Font Embedded	Whether the custom font should be embedded in the generated pdf output
Default Max Rows	The default maximum number of rows to output for a query
Specific Max Rows	The maximum number of rows to output for specific view modes defined as a comma separated list of settings with each setting in the format view-mode:value e.g. htmlGrid:5000,xls:10000. View modes are case sensitive.
RSS Link	RSS URL if ART will be used to generate RSS feeds
Mondrian Cache Expiry (hours)	Number of hours after which the mondrian cache is automatically cleared. Set to 0 to disable automatic clearing.
Date Format	Format to be used in query output for the date portion of date/datetime/timestamp fields. Format strings to be used is as per the java SimpleDateFormat class. See http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html
Time Format	Format to be used in query output for the time portion of date/datetime/timestamp fields. Format strings to be used is as per the java SimpleDateFormat class. See http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html
Scheduling Enabled	Defines whether jobs can be scheduled and whether scheduled jobs can run.
Available View Modes	The view modes that will be available to users when they run a query, defined as a comma separated list. View mode names are case sensitive and the order specified will be respected in the list shown to users. The first one is the default.

Setting	Description
Maximum Running Queries	The maximum number of queries that can be running at any one time
Maximum Pool Connections	The maximum number of connections a connection pool can open to the same datasource. Further requests are queued.
Show Results Inline	Defines whether query results are shown inline below the parameter selection box or on a new page.
Display Null Value	Defines whether the string "null" is displayed where the field in the query is null
LDAP - Server	LDAP server URL. Used for external authentication against an LDAP server.
LDAP - Authentication	Authentication method to be used with the LDAP server
LDAP - Users Parent DN	The parent DN of the user entries in the LDAP directory. Not required for Active Directory authentication.
LDAP - Realm	The LDAP realm when using a Digest-MD5 authentication method. If blank, the default realm will be used.
Windows - Domain Controller	Host name of the windows domain controller. Used for external authentication against a windows domain.
Windows - Allowed Domains	Domain name of the windows domain used for external authentication. Multiple domains can be specified, separated by commas.
Database - JDBC Driver	JDBC driver for the database used for external authentication
Database - JDBC URL	JDBC URL for the database used for external authentication
Login Page	The login page that ART will display by default. Set to Internal Login if authentication will be done by ART, or other appropriate settings if an external authentication source will be used. Setting it to Default will also use internal authentication but will display an additional page allowing for selecting the normal view or the mobile version of ART.

Once saved, these settings are stored in a file named **art.properties** in the ART_HOME\WEB-INF directory, where ART_HOME is the directory where ART is deployed.

3. Datasources

A datasource is a target database on which you want to run queries. From the **Admin Console**, use the **Datasources** option to manage datasources.

3.1. Creating a new datasource

From the Manage Datasources page, choose the ADD option and then specify the settings for the datasource. ART can execute queries against any datasource for which a JDBC driver is available.

Define Datasource	
ID	
Datasource Name	<input type="text"/>
Database Type	-- <input type="button" value="v"/> <input type="button" value="?"/>
JDBC Driver	<input type="text"/>
JDBC URL	<input type="text"/>
Username	<input type="text"/>
Password	<input type="password"/>
Connection Pool Timeout (mins)	20 <input type="button" value="v"/> <input type="button" value="?"/>
Connection Test SQL	<input type="text"/> <input type="button" value="?"/>
<input type="button" value="Submit"/>	

Attribute	Description
Datasource Name	A name to identify the datasource
JDBC Driver	JDBC driver name. If you are using a JNDI datasource, this must be blank.
JDBC URL	JDBC URL of the target database. If you are using a JNDI datasource, set this to the JNDI name of your datasource e.g. jdbc/MyDatasource. You can also use the full JNDI url e.g. java:comp/env/jdbc/MyDatasource
Username	Database user on the target database. This user should have minimum rights on the database and tables you plan to use in your queries, mostly only SELECT rights to the relevant tables.
Password	Password for the database user
Connection Pool Timeout	Sets how long a database connection can stay idle in the connection pool before being closed
Connection Test SQL	You can specify a minimal SQL query that is used to test if a connection in the pool is valid

Click on the **Submit** button. A connection test is performed to determine if the datasource details are valid and then the datasource is saved.

Note:

- The Database Type field is not saved. It is only there as a guide, to help fill the JDBC Driver and JDBC URL fields.
- From the Manage Datasources page you can also refresh and analyze the status of database connections, as well as clear the mondrian cache. When viewing datasource status, the "In Use" item indicates whether a connection is in use at that time and the last time it was used.
- You may define any number of datasources. The ART repository database itself where details of queries, users etc are stored can be a target database.
- When ART connects to a database to execute a query, a JDBC connection is established. This connection is not closed after the query finishes execution. It is maintained open to serve further requests. This reduces the overhead due to authentication when opening new connections.
- 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 **Connection 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.

3.2. 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: **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/>

Database: **Generic ODBC Datasource**

Driver Name: sun.jdbc.odbc.JdbcOdbcDriver

JDBC URL: jdbc:odbc:<dsn_name>

Note:

- The sun JDBC-ODBC bridge driver is available by default. You need to set up the DSN on the server. For optimal performance you should use the specific DBMS JDBC driver.
- ART comes with drivers for CUBRID, PostgreSQL, MySQL, HSQLDB so you don't need to download these drivers

4. Users

From the **Admin Console**, use the **Users** option to manage users.

4.1. Setting up a user

From the Manage Users page, use the ADD option to create a user.

The broad categories of users are:

- **User** - Normal User, Normal User allowed to schedule jobs
- **Low-privileged administrators** - Junior Admin (can only edit queries), Mid Admin (can manage user privileges too)
- **High-privileged administrators** - Standard Admin (can edit users and manage admin privileges), Senior Admin (can edit datasources, groups, rules etc), Super Admin (can manage everything).

Note:

- When adding a new user who will be authenticated by ART, you must specify a password, otherwise the user will not be able to log in successfully. If you are using an external authentication mechanism, you may leave the password field blank.
- When logging in, usernames are not case sensitive but passwords are case sensitive.
- It is always possible to log into ART as a Super Admin by specifying the ART repository database username and password. If the ART repository database is moved to a different server or the repository database password is changed or forgotten, you can delete the art.properties file before starting ART in order to specify the updated repository details and be able to log in. The repository data is not affected. Saving a copy of the art.properties file in a backup location before doing this is a good idea.
- It is possible to limit the datasources and query groups an administrator can deal with using the **Admin Privileges** option from the **Admin Console**.

4.2. Setting up the Public User

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

When editing a query, in the **Query Editor** page, you can see the direct URL you can use to execute a query. If the query has been granted to the public user, anyone can use this link, appending **&_public_user=true**, to run it.

For a link to a Dashboard query to work correctly without authentication, the public user needs to be given access to all the queries in the dashboard.

5. Authentication

5.1. Internal Authentication

By default, ART uses internal authentication, which authenticates users with a username and password combination defined and stored within ART. Users can change their passwords from within ART using the **Change Password** link on the Start page.

Do the following to configure ART to use internal authentication.

- Log in to ART using an admin user and go to the Settings page
- Under the Optional Settings section, set the Login Page to **Internal Login**
- Click on Submit to save these settings

5.2. External Authentication

ART can be configured to authenticate users using an external source instead of authenticating with its own username/password. This allows one to use other/existing authentication systems. The usernames that the users use to login must be defined within ART, but the password and authentication details reside within the external source and therefore the users can't change their passwords from within ART.

ART comes with support for several external authentication sources. These can be set up via the **Optional Settings** section in the Settings page.

- Microsoft Windows Domain
- LDAP
- Database

5.2.1. Windows 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. This means they don't have to remember yet another username/password combination. It also means the organisation's password policy will be in effect since authentication will be done against Active Directory.

Do the following to configure ART to use windows authentication.

- Log in to ART using an admin user and go to the Settings page
- Under the Optional Settings section, set the 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 domains to the windows domain name. Use all uppercase letters for the domain name
- Set the Login Page to **Windows Login**
- Click on Submit to save these settings

Configuration is now complete. 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. The password can be left blank when creating the user. Grant them appropriate access to queries. The users can now log in to ART using their windows username and password.

5.2.2. LDAP Authentication

Do the following to configure ART to use LDAP authentication.

- Log in to ART using an admin user and go to the Settings page
- Under the Optional Settings section, set the url for the ldap server. You can use the ldaps protocol in the url if SSL certificates are available and configured correctly
- Set the ldap authentication method to use
- Set the DN in the ldap hierarchy under which users are defined
- Set the Login Page to **LDAP Login**
- Click on Submit to save these settings

Configuration is now complete. 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). The password can be left blank when creating the user. Grant them appropriate access to queries. The users can now log in to ART using their ldap uid and password.

5.2.3. 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
- Log in to ART using an admin user and go to the Settings page
- Under the Optional Settings section, set the JDBC Driver and JDBC URL for the database for which the users have access. 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.
- Set the Login Page to **Database Login**
- Click on Submit to save these settings

Configuration is now complete. 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. The password can be left blank when creating the user. Grant them appropriate access to queries. The users can now log in to ART using their database username and password.

5.2.4. 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 showGroups.jsp page.

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

```
if ( <CONDITION> ) {  
    // Authentication Successful!  
    session.setAttribute("username", username);  
    response.sendRedirect("/art/user/showGroups.jsp?external=true");  
}
```

The user needs to be defined in ART (with or without a password). ART will check if the username exists, and if this is the case, will let the user proceed.

6. User Groups

From the **Admin Console**, use the **User Groups** option to manage user groups.

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

7. User Group Membership

Users can be added to or removed from user groups using the **User Group Membership** option in the **Admin Console**. This assignment can also be done when creating or modifying a user or when creating or modifying a user group.

8. Query Groups

From the **Admin Console**, use the **Query Groups** option to manage query groups.

All queries belong to a query group. Query groups are used to logically associate queries. When a user logs in, he selects a group and then is presented with the available queries in that group. By default, a group named "Test" is available (created by the art_tables.sql script during installation).

Note:

- You can modify the name of a group or create new groups at any time (group names need to be unique). In a standard organization, you may want to define several groups that 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 query 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 query groups 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.

9. Queries

From the **Admin Console**, use the **Queries** option to manage queries.

9.1. Introduction

An ART query is composed of a number of parts, including the following.

- **Header and Source**
The header section contains query properties like name, description, datasource etc. The source section contains the SQL statement used to retrieve the data.
- **Parameters**
Before executing a query, users may be prompted to enter or select values for query parameters. The values can be strings, numbers or dates.
- **Rules**
Rules are used to dynamically filter the query results depending on the user running the query.

9.2. Creating a query

From the **Admin Console**, click on **Queries** then click on **Create New Query** to create a query. From here you can specify the query name, description, datasource, the SQL statement for the query etc.

Define Query	
Header	
ID	Auto
Group	Art Demo ▾
Name	<input type="text"/>
Status	Active ▾ ?
Short Description (also: Title on Graphs/Dashboards)	<input type="text"/>
Description	<input type="text"/>
Contact Person	<input type="text"/>
Type	Tabular ▾ ?
Datasource	ArtRepository ▾
Uses Rules	No ▾
Show Parameters In Output	No ▾
Display Resultset	0 ?

Attribute	Description
ID	Auto-generated ID used to uniquely identify the query

Attribute	Description
Group	Query Group to which the query will be belong
Name	Name of the query as it appears to users
Status	If set to Hidden , the query will not appear to users in the Available Items list. However, it can be executed and used as normal e.g within a dashboard. If set to Disabled , the query will not appear in the Available Items list and won't run on any attempt to execute it.
Short Description	Short description for the query. For charts, this will appear as the title of the chart
Description	Longer description for the query
Contact Person	Can be used to store the name of the contact or reference person for the query
Type	The type of query
Datasource	The datasource against which the query will be executed
Uses Rules	This specifies if the query will use rules. If set to Yes , after saving the changes, a button named Rules will appear allowing you to specify the column on which to apply the rule and the rule that will be applied to the query.
Show Parameters In Output	This determines if parameter values should be displayed in the query output. If set to No , the Show Parameters checkbox in the parameters page will start as unchecked. If set to Yes , the Show Parameters checkbox will start as checked. The user can check/uncheck the Show Parameters option as desired before running the query. If set to Always , the Show Parameters checkbox will not be available to the user and parameter values will always be displayed in the query output. To display parameters when executing a query via direct URL, either add &_showParams=true to the URL, or set this field to Always .
Display Resultset	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 select statement, regardless of how many statements exist. Set to -2 to use the last 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 Multiple Statements section for more details.
X Axis Label	For charts, the x axis label
Y Axis Label	For charts, the y axis label
Chart Options	For charts, additional options that define how the chart will look
Template	For jasper reports, select the jrxml file to use. For pivot table (mondrian) queries, select the mondrian cube xml file to use. For jXLS reports, select the Excel template file to use.

Attribute	Description
XMLA URL	For mondrian via xmla or Microsoft SQL Server Analysis Services (SSAS) via xmla, the url of the xmla server
XMLA Datasource	For mondrian via xmla, the xmla datasource name as it appears in the data-sources.xml file on the xmla server
XMLA Catalog	For mondrian or mondrian via xmla, the catalog name as it appears in the mondrian cube xml file. For SSAS, the database name.
XMLA User-name	For SSAS, if the server is configured to require basic authentication, the username to use
XMLA Password	For SSAS, if the server is configured to require basic authentication, the password to use
Source	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".

9.3. Query Types

- **Tabular**

This is the default. A tabular result exportable to spreadsheet, pdf etc.

- **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
```


- **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.

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

/* output */                               /* output */
//           y-axis                         y-axis
//           |                             |
// x-axis - _ Feb Jan Mar                   x-axis - _ Jan Feb Mar
//           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.

- **Crosstab (html only)**

Same as crosstab but limited to HTML output only

- **Charts**

Display the results as a chart, exportable to pdf or png. The image can also be copied directly from the browser and pasted into any document.

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

- **XY**

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

Dynamic Series

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

- **Pie**

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

- **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
```

- **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
```

- **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
```

- **Bubble**

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

- **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
```

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**
- 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
upperBound	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.
lowerBound	Any number, positive or negative	0	Lowest value displayed
scalePos	top left bottom right	bottom	Default position of the colour scale relative to the chart
scaleLabel	Any string	None	Title the colour scale
scaleTextPos	topleft topright bottomleft bottomright	topleft	Position of the scale title relative to the colour scale

Option	Possible Values	Default	Description
scaleBorder	Hex colour code e.g. #00E000	None	Colour of border displayed around the colour scale box
stripWidth	Any positive integer	10	Width/thickness of the colour scale
subdivisions	Any positive integer	10	For grey scale or a 2 colour scheme, i.e. if no <code>color#n</code> options are configured, the number of shades of grey or shades of the 2 colour scheme to use
color#<n> e.g. <code>color#1</code> , <code>color#2</code>	<number>:<hex colour code> e.g. <code>0.0:#FF0000</code>	None	The colour to use for a given range of values. The colour is determined by <code>Value3</code> of the result set. The number is the lower bound of the range. The option value has the lower bound and colour separated by <code>:</code> e.g. to display 0-10 in red and 11-50 in green, you'll define two columns in the result set, <code>"color#1=0:#FF0000"</code> , <code>"color#2=11:#00FF00"</code>
lowerColor	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 <code>lowerColor</code> to white (#FFFFFF) and <code>upperColor</code> to red (#FF0000) if you want values to be represented as shades of red
upperColor	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 <code>lowerColor</code> to white (#FFFFFF) and <code>upperColor</code> to red (#FF0000) if you want values to be represented as shades of red

- **Group: n columns**

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" query, the query must be ordered by the the first n-1 columns.

- **Update Statement**

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

- **Text**

Used to define a piece of text that can be displayed as a standalone web page

- **Dashboard**

Used to display queries in a single portal-like page.

- **Jasper Report: Template Query**

To generate formatted reports e.g. statements given to customers, you can use Jasper Reports. To create a jasper report, use the [iReport](#) 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. Just insert the parameter labels into the SQL source section, with each parameter on a new line. 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
VARCHAR, TEXT	java.lang.String
DATE, DATETIME	java.util.Date
INTEGER	java.lang.Long
NUMBER	java.lang.Double
Multi parameters	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 query in ART, upload the main report's .jrxml file using the main template field and upload the subreport's .jrxml file using the subreport field. If there are multiple subreports, upload one subreport, save the query, upload and save the next one, etc.

- **Jasper Report: 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.

- **Pivot Table: 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.

- **Pivot Table: Mondrian XMLA**

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

- **Pivot Table: Microsoft XMLA**

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

- **jXLS Spreadsheet: 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. You can define parameters that will be passed to the jXLS template. Just insert the parameter labels into the SQL source section, with each parameter on a new line.

- **jXLS Spreadsheet: 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.

- **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
```

- **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
```

- **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.

Note:

- If the output for a query creates a file e.g. pdf, this file is stored in the **ART_HOME\export** directory

9.4. Parameters

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

- **Inline Parameters**

Inline parameters are used to pass single values to the SQL query. A "label" surrounded by the # character is used to identify the inline parameter within the SQL source. For example, the following query has two inline parameters.

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

Take the following steps to use an inline parameter

- Create the query, putting the parameter label in the appropriate place within the SQL, like in the example above
- Click on the **Save Changes** button to save the query
- Click on the **Parameters** button
- Click on the **New** button and specify **Inline** for the **Parameter Type**
- In the **Parameter Label** field, type the label exactly as it appears in the SQL source (the label text without the surrounding #).
- Specify a user friendly name for the parameter in the **Name** field
- The **Help Description** field can be used to provide additional help text for users
- Specify the data type of the parameter (integer, varchar, date etc). The parameter type will be used to validate user input. For date parameters, a date picker will be displayed for the user to select the date. Boolean parameters (true/false) are not supported.
- Specify a default value for the parameter if desired.
- If the possible parameter values should be picked from a list, set the **Use List Of Values** option to **Yes** and specify the lov query that will generate these values. This LOV query should have been created already.
- Now when a user runs the query, they will be presented with a parameters section where they can select or input values for the parameter before finally executing the query.

Note:

- The same parameter (#label#) can be used several times in the same query. All occurrences will be substituted with the same value.
- Avoid using the special labels **#rules#**, **#filter#** or **#recipient#**. These labels are used for the rules, chained parameters 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 allow input of up to 30 characters while TEXT parameters allow input of much longer character strings.
- **Workaround for boolean parameters**

There is no boolean data type for inline parameters. Different RDBMSs implement the boolean data type differently. As a workaround, you can create a static LOV query to return the boolean states true and false, as used in your database, and use that LOV query 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 an inline parameter named "boolean_param", with data type as VARCHAR, set **Use List Of Values** to **Yes** and set the LOV query to the query 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
```

- **Multi Parameters**

Multi parameters are used to pass multiple values to the SQL query. To define a multi parameter, put the WHERE...IN clause in the desired location in your query and include the parameter label 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 parameter

- Create the query, putting the WHERE...IN clause and parameter label in the appropriate place within the SQL, like in the example above
- Click on the **Save Changes** button to save the query
- Click on the **Parameters** button
- Click on the **New** button and specify **Multi** for the **Parameter Type**
- In the **Parameter Label** field, specify the parameter label as used in the SQL source e.g. "product_list" for the example above
- If the multi parameter derives its values from an LOV query, this should be created and selected in the **LOV Query** field.
- 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. Multiple values are selected using **control+click** or **shift+click**. An "All" option will be displayed to select all possible values.

Note:

- If the values for a multi 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.
- If a query that uses a multi parameter is included in a dashboard, the multi parameter will not be displayed in the parameters page for the user to modify. Multi parameters will be treated like the "All" option was selected. Only inline parameters would be available for modification before the dashboard is executed.

9.5. Executing a query via URL

It is possible to execute a query directly via URL. The view mode and parameters can be passed through the URL (all are case sensitive and special characters need to be substituted using % plus their ASCII hex values e.g. %20 for a space. See http://www.w3schools.com/tags/ref_urlencode.asp for more).

In the Query Editor page, you can see the direct URL you can use to execute the query directly (using default parameter values). This URL can be used as is, or modified to use specific parameter values if the query uses 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_user=true** must be included in the URL.

Example 1: (inline parameters)

```
http://server_name:port/art/user/ExecuteQuery?
queryId=120&viewMode=html&P_startdate=2006-04-04&P_description=%25
```

executes query 120 in html view mode.

The #startdate# inline parameter (P_startdate) is set to 2006-04-04 and the #description# inline parameter (P_description) is set to "%" (25 is the "%" character in hex so the representation in the URL is %25).

Example 2: (chart)

```
http://server_name:port/art/user/ExecuteQuery?
queryId=120&P_date=2006-04-04
```

Example 3: (group)

```
http://server_name:port/art/user/ExecuteQuery?
queryId=122&viewMode=htmlreport&SPLITCOL=2&P_param=abc
```

executes query 122 in group on 2 columns mode (SPLITCOL parameter). The first inline parameter is set to "abc".

Example 4: (pdf, public)

```
http://server_name:port/art/user/ExecuteQuery?
queryId=120&viewMode=pdf&_public_user=true
```

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

Example 5: (dashboard)

```
http://server_name:port/art/user/ExecuteQuery?
queryId=127&P_param1=value1&P_param2=value2
```

executes dashboard with id 127, setting values for inline parameters param1 and param2.

Append &_public_user=true if access without authentication is desired

Example 6: (show parameters)

```
http://server_name:port/art/user/ExecuteQuery?
queryId=120&viewMode=xls&P_startdate=2006-04-04&_showParams=true
```

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

Example 7: (multi parameters)

```
http://server_name:port/art/user/ExecuteQuery?
queryId=120&viewMode=xls&M_category=Laptops&M_category=Desktops&M_category=Tablets
```

the #category# multi parameter (M_category) has 3 values defined: Laptops, Desktops, Tablets

Example 8: (default parameter values)

```
http://server_name:port/art/user/ExecuteQuery?queryId=16
```

generates report using default parameter values

Note:

- If the query 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"  
...
```

9.6. View Modes

When running queries interactively, or scheduling them for later execution, you can specify the format in which the results should be output. When running a query via url, you can specify the view mode to be used using the `viewMode` parameter. View mode names are case sensitive. Some view modes are not available for certain types of queries.

View Mode	Display Name	Description
htmlGrid	Browser (Grid)	Shows results in a web page. The data can be sorted by clicking on the column headers. Not available for scheduled jobs.
html	Browser (Fancy)	Shows results in a web page, with minimal styling. Not available for scheduled jobs.
htmlPlain	Browser (Plain)	Shows results in a web page, with no styling.
htmlDataTable	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.
xls	Spreadsheet (xls)	Creates a Microsoft Excel spreadsheet file to download (xls format). Excel/OpenOffice/LibreOffice compatible
xlsZip	Spreadsheet (zip xls)	Creates a Microsoft Excel spreadsheet file to download (xls format), compressed using Zip.
xlsx	Spreadsheet (xlsx)	Creates a Microsoft Excel spreadsheet file to download (xlsx format). Excel/OpenOffice/LibreOffice compatible
slk	Spreadsheet (slk)	Creates a Microsoft Excel spreadsheet file to download (slk format). Excel/OpenOffice/LibreOffice compatible
slkZip	Spreadsheet (zip slk)	Creates a Microsoft Excel spreadsheet file to download (slk format), compressed using Zip.

View Mode	Display Name	Description
tsv	Spreadsheet (text tsv)	Creates a tab-separated-values text file to download
tsvZip	Spreadsheet (zip tsv)	Creates a tab-separated-values text file to download, compressed using Zip
tsvGz	Spreadsheet (gzip tsv)	Creates a tab-separated-values text file to download, compressed using GZip
pdf	Document (pdf)	Creates a pdf file to download
graph	Browser (Plain)	Shows chart in a web page. Only available for charts
pnggraph	Image (png)	Creates a png file to download. Only available for charts
pdfgraph	Chart (pdf)	Creates a pdf file to download. Only available for charts
xml		Creates an xml file
rss20		Output data as a rss2.0 compliant feed
htmlreport		Output data in grouped fashion. Only available for group queries (group on n columns query type)

You can restrict which view modes are shown to users when running queries by modifying the View Modes field in the Settings page.

Note:

- There is a limit set for the maximum number of rows that can be output with certain view modes. These limits are set, and can be modified from the Settings page.
- Use **xlsx**, **slk** or **tsv** view modes for large data sets. In particular, if you use the **xls** view mode 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.

9.7. Some Useful Queries

It may be useful to set up some queries against the ART Repository in order to monitor ART usage and performance. For example, every time a query is executed interactively or a login attempt is made, a record is created in the **ART_LOGS** table. You can create queries against this table to monitor ART usage e.g. how long it takes to execute queries, from which IP address, who attempted to login with wrong credentials etc. You can find some sample queries in the `PACKAGE_PATH\database\admin` directory.

10. Multiple Statements

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

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 select 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 allowMultiQueries=true 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

11. LOV Queries

LOV queries are used to generate parameter values for other queries. Dynamic LOV queries get their values from an sql query while static LOV queries use fixed values defined in the sql source section. An LOV query 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 the one displayed to the user on the parameters page. 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
```

11.1. Example

This section shows how to define a simple query to retrieve information about the queries stored in the ART repository. The query has one parameter (the query 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 repository
- Define a user and a query group

- **LOV Query**

This query is the one used to retrieve the list of values (i.e. the list of available queries that will be shown as a parameter). From the **Admin Console** click on the **Queries** button and then on the **Create New Query** button. Set the query name to "ART Query Names", Set the type to **LOV: Dynamic** and type the following as the SQL source:

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

Select the ART repository database as the Datasource and click on the **Save Changes** button. The query is created and the Query Editor page is now displayed. Click on the **Back to Query Management Console** button to create the main query as specified in the next paragraph.

- **Main Query**

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

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

Select the ART repository as the Datasource and click on the **Save Changes** button. The query will be created and the Query Editor page displayed. Click on the **Parameters** button, click on **New** and set the parameter type as **Inline**. Enter the label name (without #, case sensitive), a friendly name, a short description and help description. Set the **Use List Of Values** option to **Yes** and select "ART Query Names" in the **LOV Query** field. Click on the **Submit** button to save the parameter details.

Go back to the **Admin Console** and click on the **User/User Group Privileges** button. Select a user and grant him access to the query "ART Queries". Now you can log in as the user and execute the query.

Note:

- You don't have to assign access to the LOV query itself. As long as a user has access to the main query, they will be able to view the parameter values supplied by the LOV query.
- 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#
```

12. Dynamic Queries

It is possible to create Dynamic SQL queries using xml-like tags in the SQL source section. This feature allows you to modify the structure of the query based on user parameters, possibly running completely different queries depending on the user input.

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, inline 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#
```

12.1. Operators

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

12.2. Dynamic Query 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 query 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>

```

13. Drill Down Queries

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

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

13.1. Defining the main query

The main query is created like any other query. Once the query is saved, use the **Drill Down Queries** button to add or modify the drill down queries that will be available for that query.

13.2. Defining the drill down query

The drill down query is created like any other query. The only requirement is that it needs to have at least one inline parameter defined, with this parameter having the **Drill Down Column** field with a value greater than or equal to 1. The Drill Down Column refers to the column in the main query 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.

13.3. Example

- **Sales Summary** (Main Query)

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

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

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

On the Drill Down query, when defining the #region# parameter, set the option **Drill Down Column** to 1. This means this parameter will match with the first column value on the Main Query (i.e. the "Region" one):

Define Parameter	
Inline Parameter	
Parameter Label (without #, case sensitive)	<input type="text" value="region"/>
Name (User Viewable)	<input type="text" value="Region"/>
Short Description	<input type="text" value="..."/>
Help Description	<input type="text" value="..."/>
Data Type	<input type="text" value="VARCHAR"/>
Default Value (For dates use YYYY-MM-DD format. See help button)	<input type="text" value="North"/> ?
Use List Of Values (LOV) (Set to Yes if the parameter should be picked from a list)	<input type="button" value="Yes"/> <input type="button" value="No"/>
LOV Query (The query that will produce the list of values)	<input type="text" value="..."/>
Apply Rules on LOV (Filter the list of values if a rule exists for the LOV query)	<input type="button" value="Yes"/> <input type="button" value="No"/>
Slave Parameter (This parameter - slave - depends on values of another parameter - master)	<input type="text" value="No"/> ?
Drill Down Column	<input type="text" value="1"/> ?
<input type="button" value="Submit"/>	

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

On the Main Query, select the **Drill Down Queries** option in the Query Editor page

Admin Console :: Start Page :: Log Off	Logged in at 3:37 AM
----------------------------------------------------------------------------------------	----------------------

Object Editor	
<input type="button" value="Header and Source"/>	Edit header and source code
<input type="button" value="Parameters"/>	Edit Query Parameters
<input type="button" value="Drill Down Queries"/>	Edit Drill Down Queries

[Back to Object Management Console](#)

Select **New** and:

- Pick the Drill Down query from the list
- Specify the link column title (header) and link text
- Select which View Mode will be used to display the Drill Down Query results (output Format - html, xls, pdf etc). Using the **All** option will enable the user to select the view mode, and query parameters.
- Set if the result of the drill down query should be opened in a different window or in the same one

Select Drill Down Query	
Drill Down Query	Sales Details - 14 ?
Drill Down Title	Drill to details ?
Drill Down Text	Drill ?
Output Format	Browser (Grid) ?
Open In New Window	No ?
<input type="button" value="Submit"/>	

When executing the Main Query, a new column will appear on the right:

Region	Volume	Drill to details
North	2,200	Drill
South	1,800	Drill
East	1,110	Drill
West	1,200	Drill

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

Region	City	Item	Volume
North	Aba	Apples	200
North	Elto	Oranges	2,000

13.4. Using the main query's parameters

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

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

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

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

So the Drill Down query 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 Drill Down Column 1, since it's getting its value from the Main Query's column 1, while the `#city#` parameter will be set as Drill Down Column 0, since it's not getting its value from any of the Main Query's columns (the Main Query's select doesn't include the city column).

The parameter values from one query are passed down to all drill down queries down the line. So if this drill down query had another drill down query, that query can also use the value of the `#city#` parameter set in the Main Query, which is 2 queries above it in the drill down sequence.

Note:

- Multiple Drill Down queries can be linked to the same Main Query: extra columns will appear with links allowing the user to select which drill down query to run.
- A Drill Down query can have several parameters to match with any number of column values from the Main Query. It can also have inline or multi parameters that use values from the Main Query's parameters.
- If the parent query is a chart, the parameter will get the following values
 - Drill down column 1 = data value
 - Drill down column 2 = category name
 - Drill down column 3 = series name

14. Dynamic Query Datasources

When you define a query, you specify the datasource (target database) 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 queries with identical sql but different datasources, you can create one query and have the datasource as a parameter that can be selected at runtime. This eliminates the work of creating multiple queries 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 query datasources is as follows.

- From the **Admin Console**, click on the **Datasources** button and create a datasource for each of your target databases
- Create the query that will retrieve the data you want. What you select in the Datasource field is not important as the query will use a dynamically specified datasource. However, it is advisable to set this to a valid, default datasource for your sql, in case an invalid datasource is specified at runtime.
- Define an inline parameter for your query. Give the parameter a **Parameter Label** even though the sql doesn't have to contain this label, and set the **Data Type** to **DATASOURCE**. If you would like the available datasources to be selected from a list, create and specify the LOV query that will display the desired datasources.

When you run the query, it will use the value of the datasource parameter to determine which datasource to run the query on. The value of this parameter can be either a datasource id or a datasource name.

15. 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 query where column names follow RSS item element naming convention - see below. The result set column names are used as the names of the <item> sub-elements in the produced xml file.
- Grant the query to **public_user** and determine the direct URL for the query (see section on "Executing a query via URL")
- In the direct URL, set the **viewMode** parameter value to **rss20**
This URL, when executed via a browser or RSS reader/aggregator will return an RSS feed.

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

Element Name	Description
title	The item title
description	The item description
pubDate	The date where the specific item is published
link	The URL pointing to the items
guid	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>[Query Name]</title>
  <link>[Defaulted in ART settings]</link>
  <description>[Query 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.

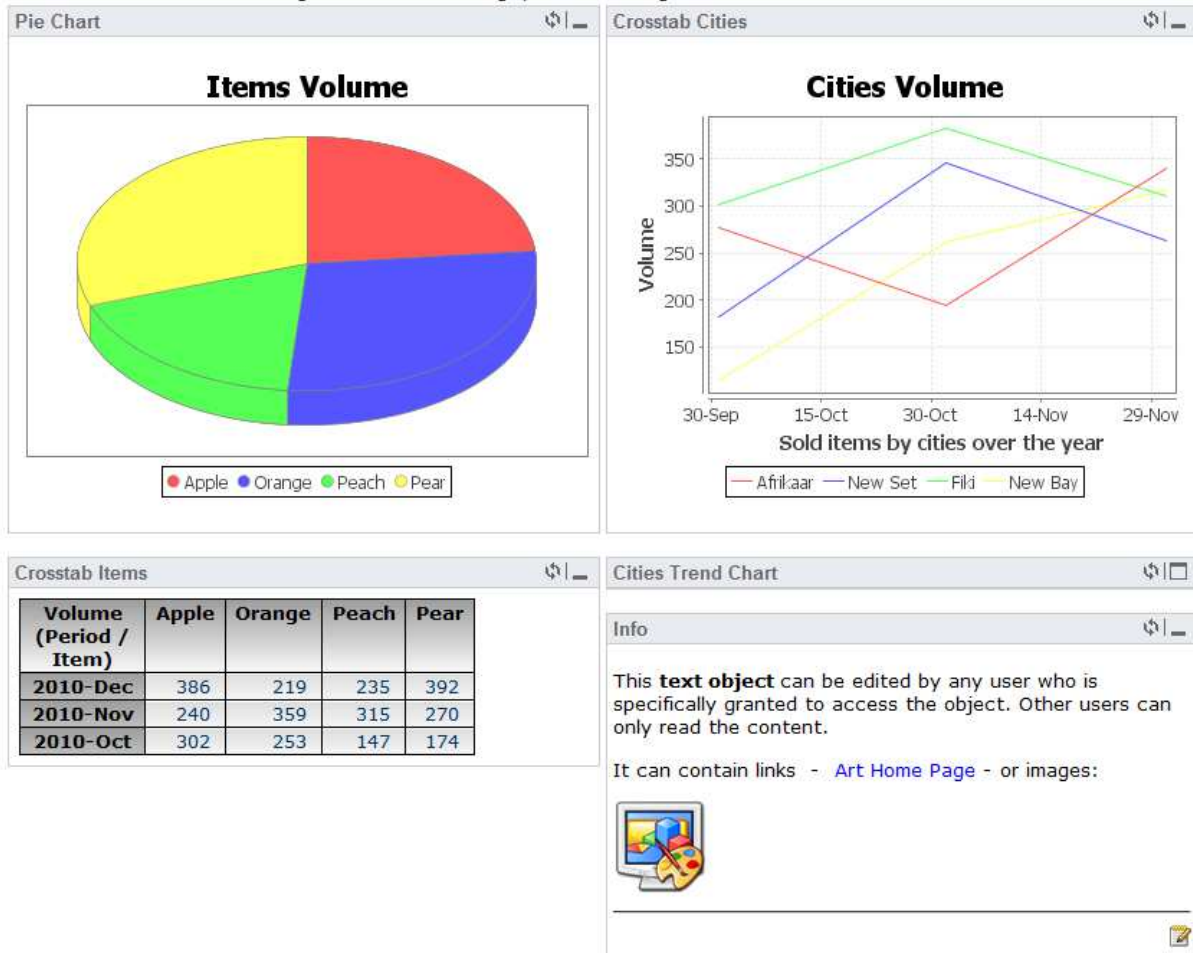
16. Dashboards

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

Any ART query 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 executes a dashboard, he can choose to modify the default parameters used by the queries in the dashboard. All the embedded queries are parsed and their parameters, if any, are displayed. If several queries use the same inline parameter label, this is displayed only once.

Regional Scorecard

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



16.1. 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 query using the query's ID (tag: <QUERYID>)
- 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>)

16.2. 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 query (tabular, chart, text) -->
<PORTLET>
<TITLE>Portlet title</TITLE>

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

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

<!-- embed ART query -->
<QUERYID>2</QUERYID>
</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 <QUERYID> tag, tabular queries are displayed using the html view mode. To override this, use the <URL> tag and type the direct access URL for the query
- 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.

17. Text Queries

A Text query is just a HTML text fragment that can be viewed either within a dashboard or as a standalone web page. When a user is viewing a text query, if he has been granted access to it, an edit link appears at the bottom right of the page. This link can be used to update the content using a WYSIWYG editor.

Note:

- Text queries are public by default i.e. anyone can view it using the direct URL. This means that you shouldn't grant explicit access to the query. If you grant access for a given user, this means that the user will be able to update the text contents e.g. if the query is granted to **public_user**, any user will be able to update its content.

18. jXLS Spreadsheets

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 spreadsheet query.

- Create an Excel file (either xls or xlsx) with any text and formatting that will be required in the final look report
- Insert jXLS syntax as appropriate to define where the raw data should go
- Create a new query in ART and select the jXLS template to use
- If the query is of type **jXLS Spreadsheet: 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 queries. In the template, use the identifier **results** to output query values. e.g.

```
${results.description}
```

- If the query is of type **jXLS Spreadsheet: Template Query**, the sql used to retrieve the data will be the one defined in the template. You can pass parameters to the template sql by typing parameter labels in the SQL source section and defining the parameters as usual. By default, the database used for the query will be the one selected in the datasource field of the query. The database to be used can also be specified within the template using either the datasource ID or datasource name. Examples.

Using the datasource defined in the query

```
<jx:forEach items="${rm.exec('select * from wishes')}" var="item">
  ${item.description}
</jx:forEach>
```

Using a specific datasource ID

```
<jx:forEach items="${rm.exec(1,'select * from wishes')}" var="item">
  ${item.description}
</jx:forEach>
```

Using a specific datasource name

```
<jx:forEach items="${rm.exec('wishlist','select * from wishes')}" var="item">
  ${item.description}
</jx:forEach>
```

19. Pivot Tables (OLAP)

ART uses the [Mondrian](#) OLAP server to provide OLAP functionality. There are 3 query types that can be created to provide OLAP analysis.

19.1. Mondrian

This query type uses ART's internal mondrian engine to execute OLAP queries. To use it, do the following.

- Create a mondrian cube schema file that defines your cube
- Create a new query in ART of the type **Pivot Table: Mondrian**
- Select the ART datasource that the cube is based on
- For the Template field, select the schema file for the cube
- In the MDX source field, enter the MDX for your query

19.2. Mondrian XMLA

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

- Create a new query in ART of the type **Pivot Table: Mondrian XMLA**
- Enter the url of the mondrian server
- Enter the required xmla datasource name. This should match exactly with the contents of the **DataSourceName** tag in the **datasources.xml** file defined on the external server
- 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
- In the MDX source field, enter the MDX for your query

19.3. Microsoft XMLA

This query type is used to access cubes defined in a Microsoft SQL Server Analysis Services (SSAS) server. To use it, do the following.

- Ensure the SSAS server is setup for HTTP access. See http://ssas-wiki.com/w/Articles#HTTP_Access for resources on how to do this.
- Ensure security access to the required database is setup. To do this,
 - Connect to the SSAS server using SQL Server Management Studio
 - Expand the required database and right-click on the **Roles** section
 - Select the **New Role** menu to create a new role
 - In the **Membership** section, add the required users. 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
 - Save the role
- Enter the url of the SSAS server
- Enter the required xmla datasource name. This is the name of the analysis services database that the cube is based on.
- If the SSAS server is configured to allow anonymous access, leave the xmla username and pass-

word 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.

- In the MDX source field, enter the MDX for your query

Note:

- You can use inline parameters in the mdx just like with any other ART query 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 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 parameter in MDX as desired...

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

- Pivot Tables use cached data where possible. To manually clear the cache, from the **Admin Console**, select the **Datasources** option, and then click on the **Clear Mondrian Cache** link. If you'd like the cache to be automatically cleared periodically, use the **Mondrian Cache Expiry** setting on the Settings page.

20. User Privileges

From the **Admin Console**, use the **User/User Group Privileges** option to define which queries a user or user group can execute.

If you give access to a query group, the user or user group will have access to all the queries in the group. If you give access to a query, the user or user group will only have access to that specific query and not any others in the query group.

You can select multiple users or user groups at the same time (using ctrl+click or shift+click) and therefore apply the GRANT or REVOKE action to a set of users or user groups.

21. Admin Privileges

From the **Admin Console**, use the **Admin Privileges** option to define which query groups and data-sources a low-privileged administrator (Junior and Mid Admin) can deal with.

You can select multiple users at the same time (using ctrl+click or shift+click) and therefore apply the GRANT or REVOKE action to a set of administrators.

22. Rules

Rules are used to filter query results. They allow the same query to be dynamically filtered depending on the user that is executing it.

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

- **Define the rule name** (it's just a label)
From the Admin Console, click on the **Rules** button and specify the rule name
- **Link the query to the rule**
On the query definition page, set the **Uses Rules** option to **Yes**.
In the SQL source section, use the special, hard coded label **#rules#** where you want the rule values to go e.g

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

The Query Editor page will now have a **Rules** button. Click on it and specify which column the rule values will apply to

- **Assign to users or user groups the rule values** that will apply to them
From the Admin Console, click on the **Rule Values** button.
Select the user or user group and rule name, and specify the rule values that will apply for that user or user group. If you want the user or user group to have multiple rule values, add each value separately. i.e Add the first value, then click on submit, then Add the next value etc.

Rule values can be EXACT values or LOOKUP to other users or user groups. You may want to create a "dummy" user in order to group rule values: for example you can create one user with username "NorthEast" and assign to him - for the rule "GeoArea" - the values NORTH and EAST. Then for every user you want to use these rule values, you can simply specify the rule type as "Lookup" and rule value as "NorthEast" (i.e. the NorthEast user).

Example:

- You have a table named "employees" with the a column named "region"
- Create a rule named "GeoArea"
- Set up a query that selects rows from the table employees 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 executes the query, 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 the string AND employees.region IN ('NORTH','EAST') to the WHERE clause).

Note:

- If a query is linked to a rule but the user who executes the query has not been set up for that rule, the query execution will not be performed and an error message will be displayed.
- If the query uses multiple rules, each user that needs to execute it must have values set for all the rules that the query uses.
- If the query uses parameters, the label **#rules#** shouldn't be used for any parameter as it will conflict with the special label 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 type is **Exact** and the rule value is **ALL_ITEMS**.

23. Chained Parameters

A parameter can be set up to display different values depending on the value selected in another parameter. The parameter that triggers the change is called the "master". The driven parameter - called "slave" - needs to be bound to a special dynamic LOV query which uses a special parameter named **filter**, and has the **#filter#** label in the WHERE clause of the SQL.

A slave parameter can be used as a master for another parameter, which in turn can have another slave and so on hence the term chained parameters.

23.1. 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 query. 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 query.

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

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

- Create a dynamic LOV query named **cities** that will display city names. The city will be a slave parameter since its value will depend on what is chosen for the country parameter. The query needs to have a parameter label named **filter**. This will be replaced at runtime with the country selected.

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

- Save the dynamic LOV query named **cities** and click on the **Parameters** button to create a new **Inline** parameter with the label **filter** and Data Type of **INTEGER**, assuming the COUNTRY_ID column holds integers. If we wanted to display cities from several countries, our cities lov would look something like

```
SELECT CITY_ID, CITY_NAME
FROM CITIES
WHERE COUNTRY_ID in(#filter#)
ORDER BY 2
```

and then we would create the filter parameter as a **Multi** parameter.

- Create the main report query. This can be any type of report. The SQL for the query will need to include inline parameter labels so that we can use the master and/or slave parameters.

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

- Once you have saved the main query, click on the parameters button to define the parameters
- Add an inline parameter with the label **country** that will be used to display the countries. Since the country values will be retrieved from an LOV query, set the **Use List Of Values** field to **Yes** and select the countries LOV query created earlier in the **LOV Query** field. Also, since this will be a master parameter and won't depend on any other parameter, leave the **Chained Parameter Sequence** field as **Not Chained**.

Define Parameter	
Inline Parameter	
Parameter Label (without #, case sensitive)	country
Name (User Viewable)	Country
Short Description	
Help Description	
Data Type	VARCHAR ▾
Default Value (For dates use YYYY-MM-DD format. See help button)	<input type="text"/> ?
Use List Of Values (LOV) (Set to Yes if the parameter should be picked from a list)	Yes ▴ No ▾
LOV Query (The query that will produce the list of values)	47 - Countries ▾
Apply Rules on LOV (Filter the list of values if a rule exists for the LOV query)	Yes ▴ No ▾
Chained Parameter Sequence (This parameter - slave - depends on values of another parameter - master)	Not Chained ▾ ?
Chained Value Position	Same as sequence ▾ ?
Drill Down Column	0 ?
<input type="button" value="Submit"/>	

- Add an inline parameter with the label **city** that will be used to display cities. Set **Use List Of Values** to **Yes** and select the cities LOV query created earlier in the **LOV Query** field. Since this parameter is a slave, set the **Chained Parameter Sequence** field to **Chain on param 1**. Each parameter can only drive one slave so you can't have two parameters both set to "chain on param 1" for example. If you need the value of a slave parameter to be determined by a parameter other than the one in the parameter sequence, set the **Chained Value Position** field as required. You can therefore have one parameter which is "chained on param 1", and the next one in the sequence "chained on param 2" (because you can't have two parameters driven by the same master), but with chained value position as "param 1" so that it's values are also determined by param 1.

Define Parameter	
Inline Parameter	
Parameter Label (without #, case sensitive)	city
Name (User Viewable)	City
Short Description	
Help Description	
Data Type	VARCHAR ▾
Default Value (For dates use YYYY-MM-DD format. See help button)	<input type="text"/> ?
Use List Of Values (LOV) (Set to Yes if the parameter should be picked from a list)	Yes ▴ No ▾
LOV Query (The query that will produce the list of values)	5 - Cities ▾
Apply Rules on LOV (Filter the list of values if a rule exists for the LOV query)	Yes ▴ No ▾
Chained Parameter Sequence (This parameter - slave - depends on values of another parameter - master)	Chain on param 1 ▾ ?
Chained Value Position	Same as sequence ▾ ?
Drill Down Column	0 ?
<input type="button" value="Submit"/>	

Parameters have a numerical order, as can be seen in the Edit Parameters page. "Chain on param 1" in this case refers to the countries parameter.

Edit Parameters	
Order	Param Type - Param Name - Label (Inline) or Column Name (Multi)
1	INLINE-Country-country
2	INLINE-City-city
<input type="button" value="Move Up"/>	
<input type="button" value="Delete"/>	
<input type="button" value="Modify"/>	
<input checked="" type="radio"/> Inline <input type="radio"/> Multi <input type="button" value="NEW"/>	
<input type="button" value=" << Back"/>	

Now when you run the query, 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 execute the query to see the final results.

Customers by City			
Country	Italy	ABC	...
City	Livorno	Country	...
View Mode	Browser (Grid)	Execute	
<input type="checkbox"/> Show Parameters			

23.2. Default Values

If you require a chained parameter to be pre-selected with certain values when the master changes, set the **Default Value** field of the parameter to a comma-separated list of values to be marked as selected by default if they exist in the new set of options for the chained parameter. Note that this list is the **actual values** of the chained parameter LOV, and not the friendly names displayed in the LOV.

Note:

- The master parameter does not need to be based on an LOV query, in which case the value typed by the user triggers the slave LOV when the master text field loses focus (e.g. by pressing the Tab key).
- Using a dynamic query, any dynamic LOV query can be converted to be used both as a normal, non-chained LOV or as a filtered LOV. In the following example the WHERE clause is ignored if the #filter# value is not available:

```
SELECT CITY_NAME FROM CITIES
<IF><EXPl>#filter#</EXPl><OP>is not null</OP>
<TEXT>WHERE COUNTRY_NAME = #filter#</TEXT>
</IF>
ORDER BY 1
```

24. Scheduling

In order to schedule a query, a user needs an access level of at least **Normal user allowed to schedule jobs**. The user's email needs to be defined correctly and a valid SMTP server needs to be specified in the Settings page if emailing is required. Once logged in, a user's scheduled jobs are available from the **My Jobs** link at the top left corner of the page. If a job has been created by another user and shared, the contents can be obtained by using the **Shared Jobs** link.

24.1. Scheduling a new job

Take the following steps to schedule a new job

- From the Start page, select the query to schedule and click on the **Next** button
- On the next page, select the **Schedule Job** option from the **View Mode** drop down list and click on the **Execute** button. If the query has parameters, you can specify the parameter values that should be used when the job runs. For date parameters, you can specify relative values e.g "ADD DAYS -1" to have the job use the previous day's date whenever it runs.
- On the next page, specify additional details about the job, including when it will run. The job name is optional, and if not specified, the query name will be used.
- Click on the **Schedule this Job** button to save the job
- The job will now run as per the schedule defined, and generate or send output as configured

Note:

- Dashboards and pivot tables cannot be scheduled

Define Job			
<i>Job Information</i>			
Job Name <input type="text"/>	Query Name: A report		
Job Type Email Output (Attachment) <input type="button" value="v"/> ?	Browser (Plain) <input type="button" value="v"/>	Enable Auditing No <input type="button" value="v"/>	Job Status Active <input type="button" value="v"/>
<i>Email Message</i>			
From	<input type="text" value="admin@localhost.local"/>		
To <small>(separate addresses with the ; character)</small>	<input type="text"/>		
Dynamic Recipients	None <input type="button" value="v"/>		
Cc	<input type="text"/>		
Bcc	<input type="text"/>		
Subject	<input type="text"/>		
Message	<div style="border: 1px solid #ccc; padding: 5px;"> <p>B I U Font family <input type="button" value="v"/> Font size <input type="button" value="v"/> HTML</p> <div style="height: 150px;"></div> </div>		
<i>Job Schedule</i>			
Nov 11, 2012 2:54:36 PM			
Saved Schedules	-- <input type="button" value="v"/>	<input type="button" value="Get"/>	
Month <input type="text"/>	Day <input type="text"/>	Week Day <input type="text"/>	
Hour <input type="text"/>	Minute <input type="text"/>		

24.2. 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 View Mode drop down menu.

- **E-Mail Output (Inline)**

The output is emailed to the specified recipients, in html format in the email body.

- **Publish**

The output is saved to a file. The file is reachable from the My Jobs or Shared 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
```

To make the alert even more useful, you can create a detailed report with the details a user may be interested in when the alert is triggered. In the message of the alert, you can include the direct URL link for the report so that when they get the email, they can click on the link and see what exactly caused the alert and perhaps take corrective action.

- **Just Run It**

Simply execute the statement (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 My Jobs or Shared 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)

Note:

- If the output for a job creates a file e.g. publish to pdf, this file is stored in the `ART_HOME\export\jobs` directory

24.3. Job 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 *Job Archives* link at the top left corner.

24.4. Job Auditing

Use the **Enable Auditing** option to enable/disable logging of additional information for a job run. If set to "No" only the last start and end time is recorded (in the ART_JOBS table). If set to "Yes", an additional record will be created in the ART_JOBS_AUDIT table indicating the start time, end time and status of each job run.

24.5. 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 save a new schedule, enter all the schedule details, tick the **Save this schedule** box and enter a name for the schedule. When the job is saved using the **Schedule this job** button, the new schedule will be saved. To reuse a schedule, select it from the **Saved 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 saved schedules using the **Schedules** option

from the **Admin Console**.

24.6. 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.

24.7. 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, set the **Allow job to be shared** field to "Yes" and select the users or user groups with whom the output will be shared. These users will be able to access the job's output from the Shared Jobs link at the top left corner. Administrators can also share jobs with users or user groups using the **Shared Jobs** option from the **Admin Console**.

24.7.1. Split Jobs

If the query 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, set the **Individualize output if query uses rules** field to "Yes". If this field is set to "No" and the query uses rules, the rule values of the job owner will be applied and all shared users will get this same output. If the query doesn't use rules, the value of this field will have no effect. A single, identical output will be available to all users.

24.8. Random Start Times

You may not care when exactly a job runs, as long as it runs in a certain time window. 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 your 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.

25. 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 queries
- 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.

25.1. Create a new Cached Result

In order to cache the results of an existing query:

- Schedule a Job for the query: select the "Schedule Job" option from the available View Modes when executing the query
- In the Define 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)
- Select the name of the table in the target datasource that will contain the data (**Cached Table name**)
- Schedule the timing when the action should take place, like any other job

Job Information	
Job Type Cache ResultSet (Append) ▼	MySQL - Test ▼
?	Cached Table Name ?
	MY_CACHED_TABLE

25.2. Accessing a Cached Result

At the given time ART will:

- Execute 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 My Jobs page.

Note:

- The Cached Table is removed when the job is deleted. However if the job is edited (via the My Jobs page) 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 inquired by other queries 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.

26. Dynamic Recipients

Dynamic recipients allows you to email query 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 query results e.g. send to users only the transactions that they created or send to a support person the open tickets they are responsible for.

Using the dynamic recipients feature will involve the following process

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

There are several ways to use dynamic recipients.

26.1. Dynamic Recipients only

If you want all the recipients to get the same data and the same email message, then define a dynamic recipients query with only one column e.g.

```
SELECT email
FROM employee
```

This column should contain the email addresses of the recipients. The name of the column doesn't matter.

26.2. 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 query 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 query 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 query list will get a personalized email with the column labels substituted with his values.

26.3. Dynamic Recipients + Filtering

If you want the recipients to get different data, you will define a dynamic recipients query 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 query like

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

This will cause the **#recipient#** label 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 query 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 labels in the email message field as desired.

27. Single Sign On

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

27.1. 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)

27.2. 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
```

27.3. 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>

```

27.4. On the client machine

- The default start page for ART needs to have been set to **Single Sign On** (done in the Settings page)
- Login to a client machine using a domain account
- Access the ART home page as usual

27.4.1. Omitting the credentials box

- **Firefox**

By default, firefox will still display a credentials box requiring a user to enter their domain user-name/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 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`

27.5. Using a keytab file

Instead of having the spnego username/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

27.6. 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
```

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** link on the top left corner of the Start Page. The most recent logs can be viewed by using the **Jump to bottom** button, or alternatively by scrolling manually to the bottom of the page. New logs are added to the bottom and you'll need to refresh the page to view them. Only administrators can view the logs. 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 comes with the [log4jdbc](#) library that enables such logging. Take the following steps to configure sql logging.

- From the **Admin Console**, select the **Datasources** option and MODIFY the datasource you're interested in
- Set the **JDBC Driver** to `net.sf.log4jdbc.DriverSpy`
- Modify the **JDBC URL** by prepending **jdbc:log4** to the existing url, e.g. resulting in a url like `jdbc:log4jdbc:mysql://localhost/mydb`
- Click on Submit to 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" />
```

- The available loggers and what events they log can be found on the log4jdbc project home page, <http://code.google.com/p/log4jdbc/>
- That's all. The logging information will now be included in the application logs when queries 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\tomcat6w.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\jdk1.6.0_21\jre\bin\server\jvm.dll (you'll need to install JDK for this)
- Increase the amount of memory available to Tomcat by setting a value in the **Maximum memory pool** textbox e.g. 512 (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 e.g. "Connection to the ART Repository is not available" when accessing ART. In particular, ART requires write access to the ART_HOME\WEB-INF, ART_HOME\WEB-INF\templates, ART_HOME\WEB-INF\hsqldb, ART_HOME\export and ART_HOME\export\jobs directories.

If Tomcat is run from a batch file

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

```
set JAVA_OPTS=-server -Xmx512m
```

29.1.2. Linux

Create a file named **setenv.sh** in the TOMCAT_HOME/bin directory (or edit 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 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, which you want to make immediately available to the users of the application.

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 or relevant sub directories under it. 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\css\art.css - Change the look of user interface items
- ART_HOME\WEB-INF\classes\art-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\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, and possibly finally on your live 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, using an unzipping tool e.g. 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 6 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
- 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\jdk1.6.0_39. 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

30.3.4. Compiling under Java 7

The artdbcp module of ART will only compile on JDK 1.6. When artdbcp code is recompiled, e.g. with a maven command to clean and regenerate all the ART modules, a JDK 1.6 compiler is needed. If you aren't using an IDE, ensure the javac file in the system path is a 1.6 version. You can use the command `javac -version` to confirm this. If you are using Eclipse, ensure the Installed JRE set as the default for the workspace points to a JDK 1.6 directory. If you are using NetBeans, use the Tools | Java Platforms menu and add a location to a JDK 1.6 directory. After that, right click on the projects one by one, select the Properties menu and select the 1.6 JDK in the Build > Compile section.

If you don't have a JDK 1.6 version installed, edit the

PACKAGE_PATH\src\art-parent\artdbcp\src\main\java\art\dbcp\Enhanced-Connection.java file by uncommenting the JDK 1.7 methods. You should then be able to compile the project using a JDK 1.7 compiler.

30.4. Custom Export Path

By default, if the output for a query creates a file e.g. pdf, this file is stored in the ART_HOME\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 on Tomcat, take the following steps.

- Create a directory named **META-INF** under the **ART_HOME** location e.g. C:\tomcat\webapps\art\META-INF
- Under this META-INF directory, create a file named **context.xml** with the following contents

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
<Environment name="REPORT_EXPORT_DIRECTORY" override="false" type="java.lang.String" value="\\server\path\"/>
<WatchedResource>WEB-INF/web.xml</WatchedResource>
</Context>
```

- Set the **value** attribute to your desired path

31. Support

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

31.1. Resources

Item	Resource
Hex colour codes	http://www.w3schools.com/html/html_colors.asp