ART Admin Manual

5.0

Table of Contents

1. Introduction

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

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

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

1.1. Administration Overview

The Administrators define a number of items including

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

The typical process when defining a report is

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

2. ART Database

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

Field	Description
Database Type	The database software that the database runs on
JNDI	Whether the ART database is a JNDI datasource
Database Protocol	An indication of the kind of syntax the database uses. This is especially important to set if using a JNDI datasource.
JDBC Driver	The JDBC driver name
URL	The database JDBC URL, or if you are using a JNDI datasource, 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	The user to use to connect to the ART database. The user needs SELECT, INSERT, UPDATE and DELETE rights on the ART tables.
Password	The database user’s password
Test SQL	A short SQL query used to determine if a connection is alive e.g. "Select 1"
Connection Pool Timeout (Mins)	How long in minutes an idle connection should be maintained in the connection pool before being closed. This setting applies to the ART database as well as all report datasources.
Max Pool Connections	The maximum number of connections a connection pool can open to the same datasource. Further requests are queued. This setting applies to the ART database as well as all report datasources.
Connection Pool Library	The connection pool library to use. This setting applies to the ART database as well as all report datasources.

3. Settings

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

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

Setting	Description
Use Anonymous Bind	Used with LDAP authentication. Defines whether to use anonymous bind when connecting to the LDAP server in order to search for and authenticate users.
LDAP Bind DN	Used with LDAP authentication. The DN to use when connecting to the LDAP server in order to search for and authenticate users.
LDAP Bind Password	Used with LDAP authentication. The password to use when connecting to the LDAP server in order to search for and authenticate users.
LDAP User ID Attribute	Used with LDAP authentication. The LDAP attribute which will be used to match ART usernames.
LDAP Authentication Method	Used with LDAP authentication. The authentication method to be used with the LDAP server
LDAP Realm	Used with LDAP authentication. The LDAP realm when using the Digest-MD5 authentication method. If blank, the default realm will be used.
Default Max Rows	The default maximum number of rows to output for a report
Specific Max Rows	The maximum number of rows to output for specific report formats, defined as a comma separated list of settings with each setting in the format `viewmode:value` e.g. `htmlGrid:5000,xls:10000`. Report formats are case sensitive.
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 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
Administrator Email	Email address which is displayed in link at the bottom of ART web pages
Date Format	Format to be used for date portions of dates. Format strings to be used is as per the Java SimpleDateFormat class.
Time Format	Format to be used for time portion of dates. Format strings to be used is as per the Java SimpleDateFormat class.
Report Formats	The report formats that will be available to users when they run a report, defined as a comma separated list. Report format names are case sensitive and the order specified will be respected in the list shown to users.

Setting	Description
Max Running Reports	The maximum number of reports that can be running at any one time
Show Header in Public User Session	Whether to show the menu bar and page footer for reports that are run by a public user
Mondrian Cache Expiry Period	Number of hours after which the mondrian cache is automatically cleared. Set to 0 to disable automatic clearing.
Scheduling Enabled	Defines whether scheduled jobs will run
RSS Link	RSS URL if ART will be used to generate RSS feeds
Max File Upload Size (MB)	Maximum file upload size. Set to -1 for no limit.
ART Base URL	The base URL for ART e.g. `http://art-server:8080/art`. This is used to include a link to publish job output in publish job reminder emails. Don't put a slash at the end.
System Locale	The locale to use in non-interactive scenarios e.g. jobs. Leave blank to use the default.
Logs Datasource	A database to which application logs should be written. The script located in the **ART_PACKAGE\database\dbappender** directory needs to have been run on the database. See the Logback DBAppender documentation for more details. The demo database already has this script run on it.
JWT Token Expiry (Mins)	The expiry time in minutes for JWT tokens generated for REST API authentication. A value of 0 means no expiry.
Enable Direct Report Emailing	Whether users can directly email generated report files
Encryption Key	Used in updating the encryption key used by ART
Error Notification To	Email address to send emails when errors occur within the application. Multiple email addresses can be specified separated by commas. Set blank to disable error notification.
Error Notification From	The "From" email address for error notification emails. The SMTP server used is the one configured within the Settings page.
Error Notification Subject Pattern	A string representing how the error notification email subject will look like. The syntax is Logback syntax.
Error Notification Level	The level for which error notification emails will be sent

Setting	Description
Error Notification Logger	The logger for which error notification emails are sent. Blank means notifications are sent for errors logged from all classes. If you would like to only receive notifications for errors occuring in jobs, you can set this to `art.jobrunners.ReportJob`. Multiple loggers can be specified separated by commas.
Error Notification Suppress After	The condition at which to suppress sending of duplicate error emails. See the Whisper documentation for details.
Error Notification Expire After	The amount of time after which duplicate error email suppression is expired. See the Whisper documentation for details.
Error Notification Digest Frequency	The frequency with which to send duplicate error email digest messages. See the Whisper documentation for details.
Minimum Password Length	The mimimum password length when a password is set from the Password page. If set to 0, this check is not done.
Minimum Lowercase Characters	The minimum number of lowercase characters in a password as set from the Password page. If set to 0, this check is not done.
Minimum Uppercase Characters	The minimum number of uppercase characters in a password as set from the Password page. If set to 0, this check is not done.
Minimum Numeric Characters	The minimum number of numeric characters in a password as set from the Password page. If set to 0, this check is not done.
Minimum Special Characters	The minimum number of special characters in a password as set from the Password page. If set to 0, this check is not done.
Options	Additional settings in JSON format

3.1. Updating the encryption key

ART uses symmetric encryption to store some fields e.g. datasource passwords. This requires the use of an encryption key, which is included in the **WEB-INF\classes\art\encryption\AesEncryptor.java** file. If you would like to change the key used to encrypt values within the application, take the following steps.

3.1.1. Using an encryption key

- Backup the application files and the ART database. If something goes wrong, it may not be possible to decrypt password fields and so they may all need to be re-entered. This not only applies to datasources but also to reports which have open/modify passwords, destinations e.g. ftp servers among others.
- Login to ART. Ensure that no other users are using the appliction when embarking on an encryption key update.
- Once you have logged in and confirmed that the application is working OK, go to the file system and modify the **WEB-INF\art-custom-settings.json** file and set the **encryptionKey** field to the key to be used. This would use AES encryption and so the key would need to be either 128 bits

(16 bytes), 192 bits (24 bytes) or 256 bits (32 bytes). If there was already a key set in this field, copy it somewhere in case something goes wrong and you need to replace it. Save the file. Do not stop the application or make this change while the application is not running. The application needs to be running already.

- Now go to the browser and on the Settings page, for the **Encryption Key** field, click on the **Update** button. If you get a success message, the update was successful. View the logs using the **View | Logs** menu to confirm that there were no errors or warnings. If there was an error, modify the **art-custom-settings.json** file and return the **encryptionKey** field the way it was before you attempted the update.
- If successful, in order to test/confirm, stop the application and then start it again. If you can run reports, the update should be successful.
- Once you have changed the encryption key, when you perform an upgrade of the application, you need to copy this key to the new version's **art-custom-settings.json** file before you first start the application. You can do this by unzipping the **art.war** file to a directory e.g. one named `art`. You can use any zip utility for this as a .war file in just a .zip file. Make the necessary change and then deploy the new directory e.g. `art` to your applicaiton server.
- In case you would like to go back to the default key, repeat this same process and set the **encryptionKey** field to the empty string `""` when updating.

3.1.2. Using an encryption password

Instead of specifying the encryption key directly, you can supply a password which will be used to generate the encryption key. Take the same steps as for using an encryption key but specify the password in the **encryptionPassword.password** field. Additionally, set the desired key length of the generated key in the **encryptionPassword.keyLength** field. The key length must be either 128, 192 or 256 as per the AES algorithm.

3.1.3. Testing string encryption/decryption

You can test or encrypt/decrypt strings used in the ART application by running the ART encryptor on the command line. To do this, open a command prompt window and navigate to the **WEB-INF\classes** folder. Use the command `java -cp "../lib/*;../etc/*;." art.encryption.AesEncryptor` on windows or `java -cp "../lib/*:../etc/*:." art.encryption.AesEncryptor` on linux. You can then supply a string to encrypt/decrypt together with appropriate parameters. An example of encrypting a string using a password may be `...AesEncryptor -e -t "clear text to be encrypted" -p "encryption password" -l 256`. You can use the help option to see available parameters e.g. `...AesEncryptor -h`.

Note:

- If you change the ART Database and that database was based on a different encryption key, you will get errors. You would therefore need to remember which encryption keys were used with which database.

4. Custom Settings

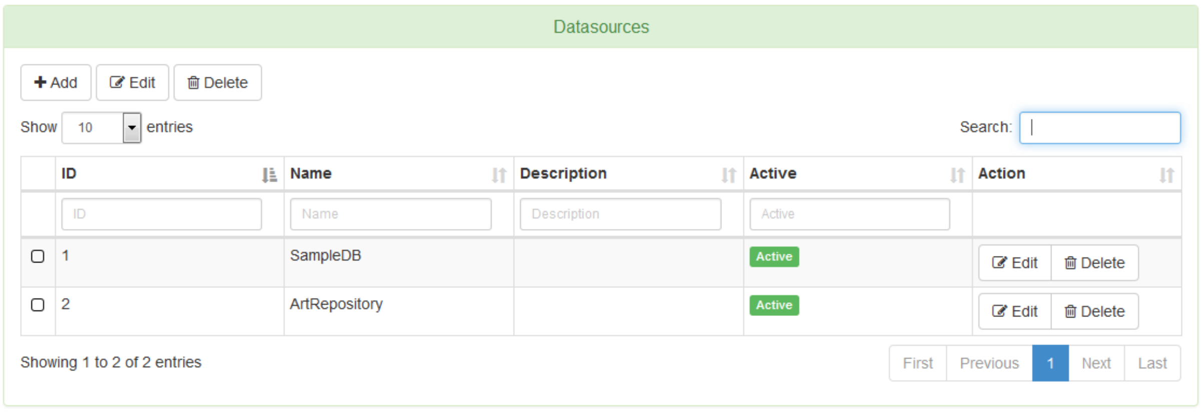
There are a number of settings that can be specified in the **WEB-INF\art-custom-settings.json** file. When settings in this file are changed, the Custom Settings cache needs to be cleared for them to take effect.

Setting	Data Type	Default	Description
showErrors	Boolean	true	Whether exception details are shown in the user interface
showErrorsApi	Boolean	true	Whether exception details are shown when making api calls
enableDirectParameterSubstitution	Boolean	true	Whether direct parameter value substitution in report sql source is allowed
exportDirectory	String		Custom directory for export files i.e. files generated by reports and jobs. If blank, these files are placed in the **WEB-INF\work\export** directory.
workDirectory	String		Custom work directory for art files e.g. templates. If blank, the **WEB-INF\work** directory is used.
checkExportFileAccess	Boolean	false	Whether export files should be checked for user access before being accessed
enableGroovySandbox	Boolean	true	Whether to apply a sandbox when running groovy scripts
enableEmailing	Boolean	true	Whether sending of emails is enabled
jwtSecret	String		String used to sign JWT tokens used in API authentication. If blank, token based api authentication is disabled.
encryptionKey	String		Key used for encryption of fields in the application e.g. datasource passwords. Must be either 16 bytes (128 bits), 24 bytes (192 bits) or 32 bytes (256 bits) as per the AES algorithm. If not specified, a default is used.
encryptionPassword.password	String		Password used to generate the key for encryption of fields in the application. Can be any length. If specified, it will take precedence over the **encryptionKey** field.

Setting	Data Type	Default	Description
encryptionPassword.keyLength	Integer	128	The desired key length when a password is used to generate the encryption key. Must be either 128, 192 or 256 as per the AES algorithm.
allowRepositoryLogin	Boolean	true	Whether to allow login using ART Database credentials

5. Datasources

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



	ID	Name	Description	Active	Action
▢	1	SampleDB		Active	Edit Delete
▢	2	ArtRepository		Active	Edit Delete

Showing 1 to 2 of 2 entries

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

5.1. Creating a new datasource

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

Field	Description
ID	An auto-generated ID used to identify the datasource
Name	A name to identify the datasource
Description	A description for the datasource
Active	Whether the datasource is available for use
Datasource Type	The type of datasource. For a **MongoDB** datasource, a MongoClient connection object will be available in the groovy script with the variable name **mongoClient**.
Database Type	The database software that the database runs on
JNDI	Whether the datasource is a JNDI datasource
Database Protocol	An indication of the kind of syntax the database uses. This is especially important to set if using a JNDI datasource.
JDBC Driver	The JDBC driver name
URL	The 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`, or the full JNDI url e.g. `java:comp/env/jdbc/MyDatasource`
Username	The database user on the target database. It is recommended that this user should have the least rights on the database and tables you plan to use in your reports, mostly only SELECT rights on the relevant tables.
Password	The password for the database user
Test SQL	A short SQL query that can be used to determine if a connection is OK e.g. `select 1`
Connection Pool Timeout (Mins)	How long in minutes an idle connection should be maintained in the connection pool before being closed
Options	Options for the datasource. The options are specified in JSON format.

Note:

- Connections do not always close gracefully - for example if a network outage occurs a broken connection might stay in the pool and would throw an error when used. The **Test SQL** query is used every **Timeout** minutes to validate the connection. If it does not run successfully, the connection is closed and removed from the pool.

5.2. Options

A number of options can be specified in the **Options** field. These are defined in JSON format with the following possibilities.

Property	Data Type	Description
limitClause	String	The syntax of a "LIMIT" clause in an SQL statement e.g. "limit {0}". This would be used when the datasource is used in a view report. The "{0}" would be replaced with the limit value at run time. This option doesn't need to be specified if the **Database Protocol** field is set.
limit	Integer	The default limit to use when the datasource is used in a view report
hikariCp	Object	Allows for specifying some HikariCP configuration options e.g. `{ "hikariCp" : { "readOnly" : true} }`. If you would like to specify options to be used for all datasources, you can specify these options in the **WEB-INF\classes\hikaricp.properties** file.

5.3. Notes

5.3.1. BigQuery (Starschema driver)

For the BigQuery (Starschema driver) database type, note the following.

- For the **Username** field, put in the google cloud service account email
- For the **Password** field, put in the path to the service account key file e.g. C:\sample\Test Project.json

5.4. Some JDBC Drivers and URLs

Database: **CUBRID**

Driver Name: cubrid.jdbc.driver.CUBRIDDriver

JDBC URL: `jdbc:cubrid:<server>:<port>:<database>[:?<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>:<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 (Connector/J 5.1), or com.mysql.cj.jdbc.Driver (Connector/J 8.0+)

JDBC URL: `jdbc:mysql://<server>[:port]/<database>[?<prop-erty>=<value>[&...]` (default port is 3306)

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

Database: **MariaDB**

Driver Name: org.mariadb.jdbc.Driver

JDBC URL: `jdbc:mariadb://<server>[:port]/<database>[?<prop-erty>=<value>[&...]` (default port is 3306)

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

Database: **PostgreSQL**

Driver Name: org.postgresql.Driver

JDBC URL: `jdbc:postgresql://<server>[:port]/<database>[?<prop-erty>=<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>[:port];database-Name=<database>[;instanceName=<instance>][;<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>[:port]/<database>[;instance=<instance>][;<property>=<value>[;...]`
(default port is 1433)

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

Database: **HSQLDB (Standalone mode)**

Driver Name: org.hsqldb.jdbc.JDBCDriver (HSQLDB 2.x) or org.hsqldb.jdbcDriver (HSQLDB 1.x)

JDBC URL: `jdbc:hsqldb:file:<file_path>[;shut-down=true;hsqldb.write_delay=false;create=false][;<property>=<value>[;...]`

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

Database: **HSQLDB (Server mode)**

Driver Name: org.hsqldb.jdbc.JDBCDriver (HSQLDB 2.x) or org.hsqldb.jdbcDriver (HSQLDB 1.x)

JDBC URL: `jdbc:hsqldb:hsql://<server>[: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>[:port]/<database>[:<property>=<value>[;...]` (default port is often either 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 or JTOpen driver)**

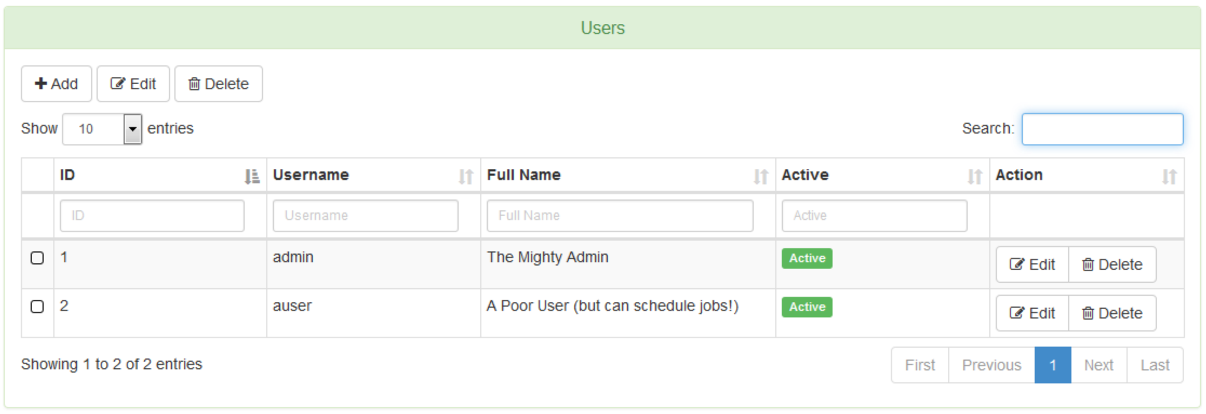
Driver Name: com.ibm.as400.access.AS400JDBCDriver

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

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

6. Users

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



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

Showing 1 to 2 of 2 entries

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

Note:

- It is possible to log into ART by specifying the ART Database username and password. This can be disabled by changing the **allowRepositoryLogin** property in the **art-custom-settings.json** file.

6.1. Creating a new user

From the Users page, use the Add button and then specify the details for the user.

Field	Description
ID	Auto-generated ID used to identify the user
Username	A username for the user, used to login to the application
Password	The password for the user. You can use the **Blank** option to specify setting of a blank password. If you have configured an SMTP server in the Settings page, you can use the **Generate and send** option to generate a random password and send an email to the user informing him of his credentials.
Full Name	The user’s full name for identification purposes
Email	The user’s email address
Description	A description for the user
Active	Whether the user account is active or disabled
Can Change Password	Whether the user can change his password
Public User	Whether the user can run reports without having logged into ART first
Access Level	The access level for the user
Default Report Group	A report group that will be pre-selected in the reports page when the user logs in
Start Report	A report ID for a report that should be run when the user logs in. Parameters can be specified after the report id e.g. `1?p-param1=value1&p-param2=value2`
User Groups	Any user groups that the user should belong to
Roles	The roles that the user has

6.2. Access Levels

Access levels affect some aspects of the of the user as they work with the application.

Access Level	**Effect**
Normal User	
Schedule User	
Junior Admin	
Mid Admin	
Standard Admin	Can work with any datasource when creating reports
Senior Admin	
Super Admin	

7. Authentication

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

7.1. Internal Authentication

Internal authentication authenticates users with a username and password combination defined and stored within ART. Users can change their passwords using the **Password** menu located under their username on the far right of the menu bar.

7.2. Windows Domain Authentication

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

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

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

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

- Set the **Allowed Windows Domains** field to the windows domain name
- ART uses the jcifs-ng library to provide windows domain authentication. You can optionally specify configuration properties for this library by modifying the **WEB-INF\classes\jcifs.properties** file. An indication of available properties can be found here.

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

7.3. LDAP Authentication

Do the following to configure ART to use LDAP authentication.

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

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

7.4. Database Authentication

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

- Have or create database users on any database. The users need to be able to connect to the database but don't need to have any other rights on the database
- In the Settings page, set the **Default Authentication Method** to **Database**
- Fill the Database Authentication JDBC fields. The JDBC driver for the database will need to be available in the **ART_HOME\WEB-INF\lib** directory or **TOMCAT_HOME\lib** directory if using Apache Tomcat.

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

7.5. CAS Authentication

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

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

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

7.6. Custom Authentication

Custom authentication code can be written e.g. executing a query on a remote database. To do this, stop the application, modify the file **CustomAuthenticationController.java** in the **art.login** package as appropriate, recompile the application by running **ant compile** from the ART home directory (ant needs to be configured), restart the application and then use a url like `http://localhost:8080/art/customAuthentication` as per your custom code.

8. Roles

Roles are used to group permissions that users have within the application. Permissions determine what users can do within the application. Use the **Configure | Roles** menu to manage roles.

8.1. Creating a new role

When creating a new role, the following fields are available

Field	Description
ID	An auto-generated ID used to identify the role
Name	A name for the role
Description	A description for the role
Permissions	The permissions included in the role

8.2. Permissions

Permissions determine the actions that users can perform within the application. A user can be assigned individual permissions directly using the **Configure | Permissions Configuration** menu, or they can get their permissions through roles assigned to them or through permissions or roles assigned to a user group they belong to.

Note:

- If you change the roles or permissions of a user, they will need to log out and log in again for the new permissions to take effect.

9. User Groups

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

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

10. User Group Membership

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

11. Report Groups

Report groups assist in assigning access rights to reports. Use the **Configure | Report Groups** menu to manage report groups.

11.1. Creating a new report group

When creating a report group, the following fields are available.

Field	Description
ID	An auto-generated ID used to identify the report group
Name	A name for the report group. It should not contain any of the following characters. `,;.`
Description	A description for the report group
Hidden	Whether the report group will be displayed in the reports page

12. Reports

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

12.1. Creating a new report

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

Field	Description
ID	Auto-generated ID used to identify the report
Name	Name of the report
Report Group	Report Group to which the report will be belong
Active	Whether the report can be run or not
Hidden	Whether the report is listed to users or not
Short Description	Short description of the report. For charts, this will appear as the title of the chart.
Description	Longer description of the report for the benefit of users
Comment	Developer comment
Contact Person	Can be used to store the name of the contact or reference person for the report
Type	The type of report
Datasource	The datasource against which the report will be executed
Uses Rules	This specifies if the report will use rules
Parameters In Output	This determines if the selected parameter values should be displayed in the report output
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.
Default Report Format	The default report format that should be selected for this report. Please note that not all report formats are available for all report types.
Omit Title Row	For tabular reports, whether the title row should be omitted in xls, xlsx, ods, slk report formats

Field	Description
Hidden Columns	For tabular reports, a comma separated list of column indices or column names for columns that should not be included in the generated output
Total Columns	For tabular reports, a comma separated list of column indices or column names for columns that should be totalled in the generated output
Date Format	For tabular reports, the format that should be used for date columns. Leave blank to use the default. The format is as per Java SimpleDateFormat format.
Number Format	For tabular reports, the format that should be used for numeric columns. Leave blank to use the default.
Column Formats	For tabular reports, the formats that should be used for specific date or numeric columns. Each column specification comes in a new line. The specification is in the format `<column index or column name>:<format>`. Examples `1:dd/MM/yyyy` or `due_date:dd MMM yyyy` or `amount:#,##0.00`. For xls and xlsx report formats, one cannot use individual column formats so the formats specified in the Date Format and Number Format fields would be used for all respective columns.
Null Number Display	For tabular reports, a string that should be used for numeric columns where the value is null. Leave blank to use the default. For xls and xlsx report formats, this setting has no effect.
Null String Display	For tabular reports, a string that should be used for string columns where the value is null. Leave blank to use the default.
Locale	For tabular reports, the locale to be used when formatting dates and numbers e.g. `de` or `en_US`. Leave blank to use the default. For xls and xlsx report formats, this setting has no effect. For fixed-width and csv report types, this determines how the date is rendered.
Fetch Size	The number of rows retrieved by the database driver per trip to the database. Leave as 0 to use the driver's default. This value reprents a trade-off between memory and query execution time and may be useful to set for reports that return large amounts of data in order to avoid out-of-memory errors. Not all databases honour this setting.
Page Orientation	The orientation to be used in generated documents e.g. with pdf output
LOV Use Dynamic Datasource	For dynamic lov reports, whether the lov should use a dynamic datasource parameter when the lov is a chained parameter
Open Password	For reports that can be exported to pdf, a password that is to be required in order to open the pdf file. In addition, for tabular reports, a password that is to be required in order to open a file with xlsx, docx, ods, odt output. Blank (empty string) passwords are not allowed and will result in no password being set i.e. the file will not request for a passwor when opening. You can use the **None** option to remove a password setting that was set previously.

Field	Description
Modify Password	For reports that can be exported to pdf, a password that can be used in order to modify certain aspects of the file (the owner password). In addition, for tabular and group reports, a password that is to be required in order to modify xlsx output. The xlsx file generated will be read-only and will require a password in order to unprotect the worksheet. Blank (empty string) passwords are not allowed and will result in no password being set i.e. the file will not request for a passwor when editing. You can use the **None** option to remove a password setting that was set previously.
Encryptor	An encryptor that should be used to encrypt report output
x-axis Label	For charts, the x axis label
y-axis Label	For charts, the y axis label
Width	For charts, the chart width
Height	For charts, the chart height
Background Colour	For charts, the background colour
y-axis Min	For charts, the minimum value for the y-axis. Leave as 0 to use the full data range.
y-axis Max	For charts, the maximum value for the y-axis. Leave as 0 to use the full data range.
Rotate x-axis labels at	For charts, the number of categories at which the labels will be displayed vertically instead of horizontally. Set to 1 to always display labels vertically.
Remove x-axis labels at	For charts, the number of categories at which labels are omitted from the chart. Set to 1 to always omit labels.
Secondary Charts	For charts, a comma separated list of report IDs of other charts which are to be displayed with this one. These charts will use separate y-axes. Some chart types cannot have secondary charts, including pie charts, speedometer, bubble and heat map charts.
Template	The template file to use for the report
XMLA Datasource	For mondrian via xmla, the xmla datasource name as it appears in the **datasources.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.
Source Report	A report ID of a report which should be used to provide the report source for this report
Link	For the link report type, the link to open. If it is an external url, it needs to start with the protocol section e.g. `http://` or `https://`.
Open In New Window	For link reports, whether the link will open in a new window when clicked from the ART home page

Field	Description
Max Running Reports	The maximum number of concurrent runs allowed for this report. A value of 0 means no maximum.
Max Running Reports Per User	The maximum number of concurrent runs allowed for this report per user. A value of 0 means no maximum.
Use Groovy	Whether the report source is groovy rather than SQL
Saved Options	Dynamically generated options e.g. when saving the state of a PivotTable.js report
Options	Provides additional options depending on the report type. The options are specified in JSON format with different report types expecting different JSON specifications.
Source	The SQL query used to retrieve the required data

Note:

- 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".

12.1.1. Tags

Some special strings or tags can be used within the SQL statement and are substituted at runtime with their respective values.

Tag	Description
`: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)
`:reportId:`	Replaced by the report id
`:jobId:`	Replaced by the job id if running from a job

12.2. Report Types

12.2.1. Tabular

A tabular result exportable to spreadsheet, pdf etc.

12.2.2. Tabular (html only)

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

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

12.2.3. Crosstab

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

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

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

OR

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

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

```
/* 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.

12.2.4. Crosstab (html only)

Same as crosstab but limited to HTML output only

12.2.5. Tabular: Heatmap

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

Property	**Data Type**	**Default**	**Description**
columns	Integer[]		Indexes of the columns that should be coloured, with the first column having an index of 1. If not specified, all columns are included in the colouring. If any cell in the applicable columns contains a non-numeric value, a javascript error will result, and will be indicated in the browser console.
colors	String[]		Colours that should be applied, beginning with colours for the lower values going up. The colour strings are specified in hex format e.g. `#ffffff`. For example for a two colour scheme in a red range, one can specify the setting as `"colors": ["#ffffff", "#ff4a53"]`
perColumn	Boolean	false	If true, each applicable column is graded and coloured separately. If false, the whole table is considered when grading the colours.

12.2.6. Charts

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

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

12.2.7. XY

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

```
Dynamic Series
```

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

12.2.8. Pie

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

12.2.9. Bars/Stacked Bars/Line

Static Series

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

Dynamic Series

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


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

12.2.10. Time/Date Series

Static Series

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

Dynamic Series

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


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

12.2.11. Speedometer

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


Example:
SELECT reading, 0, 100, "degrees",
"50:#00FF00:Normal",
"80%:#FFFF00:Warning",
"100:#FF0000:Critical"
FROM temperature_reading
```

12.2.12. Bubble

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

12.2.13. Heat Map

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


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

12.2.14. Chart Options

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

Property	Data Type	Default	Description
seriesColors	Object		Defines specific colours to be used for given chart series. Each property of the object consists of the series ID as the property name and the hex colour code as the property value. Multiple properties can be specified e.g. `"seriesColors": {"0" : "#ff4a53", "2" : "#4a538a"}`. The first series has an ID of 0.
dateFormat	String		For date series and time series charts, defines the date format to be used on the x-axis, overriding the one automatically assigned by the chart.
dynamicSeries	Boolean	false	Whether the data represents a dynamic series chart
itemLabelAnchor	String	OUTSIDE12	Determines the positioning of data label values. Possible values is as per the JFreeChart ItemLabelAnchor class.

Note:

- ART uses the cewolf and jfreechart libraries to generate charts. These libraries in turn use standard java AWT to plot charts. In order to work correctly, AWT needs a graphic display. If you are using a "headless" workstation (i.e. a Unix box without X) you need to start the JVM with the option **-Djava.awt.headless=true**

- You can specify specific colours to be used for the series displayed in a chart by adding columns to the query resultset in the format `seriesColor:<series index>:<hex color code>` e.g. for a pie chart,

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

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

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

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

Option	Possible Values	Default	Description
upperBound	Any number, positive or negative	1	The 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	The lowest value displayed
scalePos	top left bottom right	bottom	The default position of the colour scale relative to the chart
scaleLabel	Any string		The title of the colour scale
scaleTextPos	topleft topright bottomleft bottomright	topleft	The position of the scale title relative to the colour scale
scaleBorder	Hex colour code e.g. `#00E000`		The colour of the 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 `color#n` options are configured, the number of shades of grey or shades of the 2 colour scheme to use

Option	Possible Values	Default	Description
`color#<n>` e.g. `color#1`, `color#2`	`<number>:<hex colour code>` e.g. `0.0:#FF0000`		The colour to use for a given range of values. The colour is determined by `Value3` of the result set. The number is the lower bound of the range. The option value has the lower bound and colour separated by `:` e.g. to display 0-10 in red and 11-50 in green, you'll define two columns in the result set, `"color#1=0:#FF0000"`, `"color#2=11:#00FF00"`
lowerColor	Hex colour code		The colour of the lowest value if you want a 2 colour scheme with a linear gradient from the lower colour to the upper colour e.g. set `lowerColor` to white (`#FFFFFF`) and `upperColor` to red (`#FF0000`) if you want values to be represented as shades of red
upperColor	Hex colour code		The colour of the highest value if you want a 2 colour scheme with a linear gradient from the lower colour to the upper colour e.g. set `lowerColor` to white (`#FFFFFF`) and `upperColor` to red (`#FF0000`) if you want values to be represented as shades of red

12.2.15. Group

Groups the report data.

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

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

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

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

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

12.2.16. Update Statement

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

12.2.17. Text

Used to define a piece of text that can be displayed

12.2.18. Dashboard

Used to display reports in a single portal-like page

12.2.19. Dashboard: Gridstack

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

12.2.20. JasperReports: Template Query

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

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

ART parameter type	JasperReports parameter type
Varchar, Text	java.lang.String
Date, DateTime	java.util.Date
Integer	java.lang.Integer
Double	java.lang.Double
Multi-Value 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 report in ART, upload the main report’s .jrxml file using the main template field and upload the subreport’s .jrxml file using the resources field. If there are multiple subreports, you can select multiple files and upload all of them at once using the **Start upload** button. You can also select files by dragging from the file explorer and dropping to the browser page. Once you have uploaded the subreports, also specify the file names in the **Options** field using the **subreports** property that takes a list of strings e.g. `{"subreports": ["subreport1.jrxml", "subre-port2.jrxml"]}`.

12.2.21. JasperReports: ART Query

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

12.2.22. JPivot: Mondrian

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

12.2.23. JPivot: Mondrian XMLA

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

12.2.24. JPivot: Microsoft XMLA

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

12.2.25. Jxls: Template Query

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

12.2.26. Jxls: ART Query

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

12.2.27. FreeMarker

Displays a report in the browser based on a FreeMarker template. The query as defined in the SQL source will be used to generate the results. The data can be accessed in the template by iterating over the **results** variable. If you are sure a particular column will never have null values, you can use the simple dot notation to access it e.g. **${result.description}** to access a column in the result set named "description". If the column can have null values, use the default-value operator "!" after the column name e.g. **${result.description!}**. (See http://freemarker.org/docs/dgui_template_exp.html#dgui_template_exp_missing). Report parameters are also available using the parameter name e.g. **${param1.value}**. Variables named **contextPath** and **artBaseUrl** are available e.g. to include css or javascript files. **artBaseUrl** is only available if it is set in the **Settings** page and **contextPath** is only available for interactive reports and is not available with jobs. An example report is included in the demo database.

A number of options can be specified in the **Options** field.

Property	Data Type	Default	Description
extension	String	txt	The extension to use for generated file names with the **file** and **fileZip** report formats

12.2.28. Velocity

Displays a report in the browser based on a Velocity template. The query as defined in the SQL source will be used to generate the results. The data can be accessed in the template by iterating over the **results** variable. If you are sure a particular column will never have null values, you can use the simple dot notation to access it e.g. **${result.description}** to access a column in the result set named "description". If the column can have null values, use quiet reference notation e.g. **$!{result.description}**. (See https://velocity.apache.org/engine/2.0/user-guide.html#quiet-reference-notation). Report parameters are also available using the parameter name e.g. **${param1.value}**. Variables named **contextPath** and **artBaseUrl** are available e.g. to include css or javascript files. **artBaseUrl** is only available if it is set in the **Settings** page and **contextPath** is only available for interactive reports and is not available with jobs. Also a NumberTool object named **numberTool** is available for formatting numbers e.g. **$numberTool.format('#,##0.00',$result.amount)**. Similarly, a DateTool object named **dateTool** is available for formatting dates. An example report is included in the demo database.

A number of options can be specified in the **Options** field.

Property	Data Type	Default	Description
extension	String	txt	The extension to use for generated file names with the **file** and **fileZip** report formats

12.2.29. Thymeleaf

Displays a report in the browser based on a Thymeleaf template. The query as defined in the SQL source will be used to generate the results. The data can be accessed in the template by iterating over the **results** variable e.g. **${result.description}** to access a column in the result set named "description". Report parameters are also available using the parameter name e.g. **${param1.value}**. Variables named **contextPath** and **artBaseUrl** are available e.g. to include css or javascript files. **artBaseUrl** is only available if it is set in the **Settings** page and **contextPath** is only available for interactive reports and is not available with jobs. An example report is included in the demo database.

A number of options can be specified in the **Options** field.

Property	Data Type	Default	Description
extension	String	txt	The extension to use for generated file names with the **file** and **fileZip** report formats

12.2.30. ReactPivot

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

12.2.31. PivotTable.js

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

12.2.32. PivotTable.js: CSV Local

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

12.2.33. PivotTable.js: CSV Server

Generates a pivot table based on data from a specific csv file uploaded to the server. This file can be uploaded using the **Add files** button. The file name will also need to be specified in the **Options** field as follows.

Property	Data Type	Description
dataFile	String	The name of the csv file e.g. `myfile.csv`

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

12.2.34. XDocReport: FreeMarker engine - Docx

You can use MS Word (docx), LibreOffice Writer (odt) or MS PowerPoint (pptx) docments as a template for reports. You simply create the document as you would want it to appear and then insert fields that will be replaced with data at runtime. This uses the XDocReport library. You can have a look at the XDocReport wiki for examples on how to create templates for different needs. The query results will be contained in a variable named **results** which can be used in the FreeMarker or Velocity expressions. You can also use an object named **jdbc** to run queries that you define within the template. This works the same way as with Jxls Template Query reports. See the **Jxls Reports** section for more details.

Displays a report based on the selected docx template, which uses FreeMarker syntax for expressions within the document. The query as defined in the SQL source will be used to generate the results.

12.2.35. XDocReport: Velocity engine - Docx

Displays a report based on the selected docx template, which uses Velocity syntax for expressions within the document. The query as defined in the SQL source will be used to generate the results.

For velocity templates you can use syntax like **$!result.description** in case a field may contain null values. Also for velocity templates, a NumberTool object named **numberTool** is available for formatting numbers e.g. **$numberTool.format('#,##0.00',$result.amount)**. Similarly, a DateTool object named **dateTool** is available for formatting dates.

12.2.36. XDocReport: FreeMarker engine - ODT

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

12.2.37. XDocReport: Velocity engine - ODT

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

12.2.38. XDocReport: FreeMarker engine - PPTX

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

12.2.39. XDocReport: Velocity engine - PPTX

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

12.2.40. 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
```

If using groovy data, return a List of the values to be used, or if you require the value and display text to be different, return a Map whose keys are the values to be used and whose value is a String representing the display text.

12.2.41. 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
```

12.2.42. 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.

12.2.43. Dygraphs

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

12.2.44. Dygraphs: CSV Local

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

12.2.45. Dygraphs: CSV Server

Generates a dygraphs chart based on data from a specific csv file uploaded to the server. This file can be uploaded using the **Add files** button. The file name will also need to be specified in the **Options** field as follows.

Property	Data Type	Description
dataFile	String	The name of the csv file e.g. `myfile.csv`

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

12.2.46. DataTables

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

Property	Data Type	Default	Description
outputDateFormat	String		The format of output data for date columns. If blank, the data will be output as is. The format is as per the Java SimpleDateFormat format.
outputDateTimeFormat	String		The format of output data for datetime columns. If blank, the data will be output as is. The format is as per the Java SimpleDateFormat format.
showColumnFilters	Boolean	false	Determines if column filters will be displayed
dtOptions	Object		Options related to the DataTables output. Available options are as per the DataTables documentation. Where a `<th>` column class name is required, you can use `rcol-<column name>`. Spaces and other characters apart from numbers, letters, underscores and hyphens will be replaced with a hyphen; so for a report column named `Due Date`, the class name to use would be `rcol-Due-Date`.

12.2.47. DataTables: CSV Local

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

12.2.48. DataTables: CSV Server

Enables display of data from a specific csv file uploaded to the server. This file can be uploaded using the **Add files** button. The file name will also need to be specified in the **Options** field as follows.

Property	Data Type	Description
dataFile	String	The name of the csv file e.g. `myfile.csv`

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

12.2.49. Fixed Width

Generates fixed width output. The **Options** field determines the formatting of the output and can have the following properties.

Property	Data Type	Default	Description
fieldLengths	Integer[]. **required**		Used to define the field lengths of the output e.g. `[5,6]` means that the first field will have a length of 5 and the second field will have a length of 6
fieldLengthsByName	Object[]. **required**		Each object specifies the field name and the field length e.g. `[{"order_id": 8}, {"order_date": 14}]` means the `order_id` field will have a length of 8 and the `order_date` field will have a length of 14. You must specify either the **field-Lengths** or the **fieldLengthsByName** prop-erty.
includeHeaders	Boolean	true	Determines whether the column names will be included in the output as the first row
defaultAlignmentForHeaders	String		Either **left**, **right** or **center**. If present, over-rides any field alignment set and uses the given alignment for the header columns
padding	String	space	A single character defining the default padding to use for fields. Must not be the empty string.
useDefaultPaddingForHeaders	Boolean	true	Determines if the headers should use the global **padding** setting or use the padding set for individual fields
dateFormat	String		Determines the format that should be used for DATE columns
dateTimeFormat	String		Determines the format that should be used for DATETIME columns
numberFormat	String		Determines the format that should be used for numeric columns
fieldDateFormats	Object[]		Each object specifies the date format and an array of fields that should use that format e.g. `[{"dd/MM/yyyy": ["order_date", "due_date"]}, {"dd-MMM-yyyy": ["payment_date"]}]`

Property	Data Type	Default	Description
fieldNumberFormats	Object[]		Each object specifies the number format and an array of fields that should use that format e.g. `[{"0.00": ["amount"]}, {"#,##0.#": ["volume"]}]`
fieldAlignmentByName	Object[]		Each object specifies the alignment to be used and an array of field names that should use that alignment e.g. `[{"right": ["amount", "volume"]}]`. The alignment is either **left**, **right** or **center**.
fieldAlignmentByPosition	Object[]		Each object specifies the alignment to be used and an array of field positions that should use that alignment e.g. `[{"right": [2, 5]}]`. The alignment is either **left**, **right** or **center**. The first field in the output is marked as position 0.
fieldPaddingByName	Object[]		Each object specifies the padding to be used and an array of field names that should use that padding e.g. `[{"_": ["amount", "volume"]}]`. The padding should be a single character.
fieldPaddingByPosition	Object[]		Each object specifies the padding to be used and an array of field positions that should use that padding e.g. `[{"_": [2, 5]}]`. The padding should be a single character. The first field in the output is marked as position 0.

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

12.2.50. CSV

Generates csv output. The **Options** field can be used to modify the format of the output and it can have the following properties.

Property	Data Type	Default	Description
delimiter	String	,	The delimiter to use
quote	String	"	The character to use to quote fields in which the delimiter appears
quoteAllFields	Boolean	false	Whether all the data should be quoted, whether the data contains the delimiter or not

In addition to these properties, the following properties available with fixed width reports can also be used, having the same configuration and effect as in fixed width reports: **includeHeaders**, **dateFormat**, **dateTimeFormat**, **numberFormat**, **fieldNumberFormats**, **fieldDateFormats**.

12.2.51. C3.js

Generates charts using the C3.js library. The demo database contains some sample reports of this type. A number of options can be specified in the **Options** field. Also, a javascript file containing chart configuration can be uploaded using the **Template** field.

Property	Data Type	Default	Description
cssFile	String		The file name of a css file to use to provide custom styling e.g. `mycss.css`. This file can be uploaded using the **Add files** option.
chartTypes	String[]		Chart types that should be available for dynamic changing of the chart type. Possible values include **line**, **spline**, **step**, **area**, **areaSpline**, **areaStep**, **bar**, **scatter**, **pie**, **donut**, **gauge** or **all**.
x	String		For bar and line charts, the column to use as the x-axis category
value	String[]		The columns to use for data values
type	String		The initial chart type. Possible values include **line**, **spline**, **step**, **area**, **area-spline**, **area-step**, **bar**, **scatter**, **pie**, **donut**, **gauge**.
template	String		An optional file name of a javascript file that contains chart configuration. If the file was selected using the **Template** field, this is not needed.
groupedTooltip	Boolean	true	Whether to show grouped tooltips
showLegend	Boolean	true	Whether to show the legend
rotatedAxis	Boolean	false	Whether the axis should be rotated. For bar charts, setting to **true** will result in a horizontal bar chart.
showTooltip	Boolean	true	Whether tooltips should be shown
legendPosition	String		The legend position. One of **bottom**, **right**, **inset**.
width	Integer		The width of the chart in pixels
height	Integer		The height of the chart in pixels
xAxisLabel	String		The x axis label
yAxisLabel	String		The y axis label
xAxisLabelPosition	String		The x axis label position. See http://c3js.org/reference.html#axis-x-label
yAxisLabelPosition	String		The y axis label position. See http://c3js.org/reference.html#axis-x-label
groups	String`[][]`		Description of columns that should be grouped e.g. to get a stacked bar chart. See http://c3js.org/reference.html#data-groups

12.2.52. Plotly.js

Generates charts using the plotly.js library. A number of options can be specified in the **Options** field. Also, a javascript file containing chart configuration can be uploaded using the **Template** field.

Property	Data Type	Default	Description
template	String		An optional file name of a javascript file that contains chart configuration. If the file was selected using the **Template** field, this is not needed.
barmode	String		Set to **stack** to get a stacked bar chart
xColumn	String		The column to use as the x-axis category
yColumns	String[]		The columns to use for data values
type	String		The plotly type e.g. pie, scatter, bar
mode	String		The plotly mode e.g. lines, lines+markers
chartTypes	String[]		Chart types that should be available for dynamic changing of the chart type. Possible values include **line**, **bar**, **scatter**, **pie**, **donut**.
title	String		The chart title
xAxisTitle	String		The x axis title
yAxisTitle	String		the y axis title
showLegend	Boolean	true	Whether to show the legend
showText	Boolean	false	For bar charts, whether the values should be shown on the bars
textPosition	String		With **showText** as **true**, the position of the values. Sets the plotly **textposition** property.
width	Integer		The width of the chart in pixels
height	Integer		The height of the chart in pixels
hole	Double (0.0 - 1.0)		For donut charts, the size of the hole in the middle of the chart
bundle	String		Use **cartesian** to use charts available in the cartesian bundle. See https://github.com/plotly/plotly.js/tree/master/dist
orientation	String		Use **h** to get horizontal bar charts
hoverInfo	String		Determines what is shown in tooltips. Sets the plotly **hover-info** property.

Property	Data Type	Default	Description
pieValueColumn	String		For tabular reports, the column that should be used for pie chart values. If not specified, the last numeric column in the data will be used.

12.2.53. Chart.js

Generates charts using the Chart.js library. A javascript file containing chart configuration needs to be uploaded using the **Template** field. The demo database contains some sample reports of this type. A number of options can be specified in the **Options** field.

Property	Data Type	Default	Description
width	Integer	300	The width of the chart in pixels
height	Integer	100	The height of the chart in pixels

12.2.54. Datamaps

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

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

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

12.2.55. Datamaps: File

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

12.2.56. Leaflet

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

Property	Data Type	Default	Description
height	String	400px	CSS height for the div that contains the map
cssFile	String		The file name of a css file that contains custom css to be used with the display. This file can be uploaded using the **Add files** option.
dataFile	String		The file name of a file that contains extra data to be used with the map e.g. a GeoJSON file. This file can be uploaded using the **Add files** option.
jsFiles	String[]		File names of javascript files that need to be incorporated for the map display e.g. leaflet plugins. The files can be uploaded using the **Add files** option.
cssFiles	String[]		File names of additional css files to be used with the display. The files can be uploaded using the **Add files** option.

12.2.57. OpenLayers

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

12.2.58. Saiku: Connection

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

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

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

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

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

12.2.59. Saiku: Report

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

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

12.2.60. OrgChart: Database

Displays an organisation chart using data from a database. It makes use of the OrgChart library to display the org chart. The database query must have at least a column named **id**, another named **parent_id**, one named **name** and another named **title**. These names are case sensitive. The id column is a unique id among the returned records. It would typically represent a position. The parent_id column refers to the id that is the parent of the given record. The name column would typically be an employee name and appears at the top half of an org chart node. The title column would typically be the role or job title for the employee and would appear at the bottom half of an org chart node. The demo database contains a sample report of this type.

A javascript file containing chart configuration can be set/uploaded using the **Template** field. This file can be used to extend the **orgChartSettings** variable e.g. to set the **nodeTemplate** or **createNode** property. There are also a number of options that can be specified in the **Options** field.

Property	Data Type	Default	Description
cssFile	String		The file name of a css file that contains custom css to be used with the org chart e.g. `"org.css"`. This file can be uploaded using the **Add files** option.
nodeTitle	String	name	The name of the field that will be displayed in the top half of a node
nodeId	String	id	The name of the field that is a unique identifier among all the records
toggleSiblingsResp	Boolean	false	Whether users can show/hide left/right sibling nodes respectively by clicking left/right arrow.
depth	Integer	999	The level that the org chart is expanded to when it is shown initially
exportButton	Boolean	false	Whether the export button is displayed
exportFilename	String	OrgChart	The name of the file used when exporting
exportFileextension	String	png	Either **png** of **pdf**. Determines whether the export will be to png or pdf
parentNodeSymbol	String	fa-users	The icon to be displayed for nodes that have children nodes under them
draggable	Boolean	false	Whether users can drag & drop the nodes. Note that this feature doesn’t work on IE
direction	String	t2b	How the hierarchy is rendered. One of **t2b** ("top to bottom"), **b2t** ("bottom to top"), **l2r** ("left to right") or **r2l** ("right to left").
pan	Boolean	false	Whether users can pan the org chart by mouse drag&drop
zoom	Boolean	false	Whether users can zoom in or out using the mouse
zoominLimit	Double	7	The zoom in limit
zoomoutLimit	Double	0.5	The zoom out limit
nodeContent	String	title	The name of the field that will be displayed in the bottom half of the node. Set to **null** or the empty string to have this section omitted from nodes.

12.2.61. OrgChart: JSON

Displays an organisation chart using data from a JSON definition specified in the report source section. The same options applicable with the **OrgChart: Database** report type are also applicable. The demo database contains a sample report of this type.

12.2.62. OrgChart: List

Displays an organisation chart using data from a html unordered list definition specified in the report source section. The same options applicable with the **OrgChart: Database** report type are also applicable. The demo database contains a sample report of this type.

12.2.63. OrgChart: Ajax

Displays an organisation chart using data from a url specified in the report source section. The same options applicable with the **OrgChart: Database** report type are also applicable.

12.2.64. ReportEngine

Generates tabular or pivot table output using data from a database. It uses the ReportEngine library. A number of options can be specified in the **Options** field.

Property	Data Type	Default	Description
groupColumns	Object[]		An array of group column definition objects. Each object has the following properties. **id -** a string for the column index or name of the group column, **showDuplicateValues** - a boolean for whether duplicate values should be displayed, **index -** an integer for the column index when used with the **ReportEngine: File** report type.
showTotals	Boolean		Whether sub-totals should be displayed when using group columns
showGrandTotal	Boolean		Whether the grand-total should be displayed when using group columns
sortValues	Boolean	false	Whether values are already sorted in the input data. When using group columns, the data must be sorted. If it doesn't come in as sorted, you can set this property to **true**.
dataColumns	Object[]		An array of data column definition objects. Each object has the following properties. **id -** a string for the column index or name of the data column, **calculator -** a string representing a calculator to use if a group column is in use. The string would be one of the following (case sensitive). **SUM**, **COUNT**, **AVG**, **MIN**, **MAX**, **FIRST**, **LAST**. **calculatorFormatter -** a string to determine how the calculator value should be displayed. This would be according to Java **String.format()** syntax e.g. "%,.2f"
pivot	Boolean	false	Whether to output the data as a pivot table
pivotHeaderRows	String[]		For pivot table output, column index or column names of columns to be used as pivot table header rows. For **ReportEngine: File** report types, this must be column indices.
pivotData	data column object		For pivot table output, the definition of the column that will provide the data for the pivot table
separator	String	,	For **ReportEngine: File** report types, the field separator used within the file
firstLineIsHeader	Boolean	true	For **ReportEngine: File** report types, whether the first line in the file is a header row
url	String		For **ReportEngine: File** report types, if the data is coming from a url, the url to the data file

A note that pivot table output may not display correctly with pdf, docx, odt report formats.

12.2.65. ReportEngine: File

Generates tabular or pivot table output using data from a file. It uses the ReportEngine library. A number of options can be specified in the **Options** field, with the available options being similar to those available for the **ReportEngine** report type.

12.2.66. MongoDB

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

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

Example using MongoDB Java Driver syntax

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

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

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

return records;
```

Example using Jongo syntax

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

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

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

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

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

return records;
```

Example using Morphia syntax

```
package art.groovy

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


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

Morphia morphia = new Morphia();

morphia.map(Employee.class);

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

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

return records;
```

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

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

Property	Data Type	Default	Description
columns	String[]		Names of columns that should be included in the tabular output. This may be useful when querying documents that may have different attributes. If not specified, ART will use the first document retrieved to determine the columns that will be included in the output.
columnDataTypes	Object[]		Each object specifies a column name and the data type that is to be used for that column e.g. `[{"col1": "numeric"}]`. The possible data types are **string**, **numeric**, **date** and **datetime**. The default is `string` so if a column is to be treated as a general string, it doesn't need to be specified.
columnLabels	Object[]		Each object specifies a column name and the heading to be used for that column e.g. `[{"col1": "Column 1"}]`. If not specified, the column name is used as the heading.
outputDateFormat	String	dd-MMM-yyyy	The format of output for columns with a **date** specification. The format is as per Java SimpleDateFormat format.
outputDateTimeFormat	String	dd-MMM-yyyy HH:mm:ss	The format of output for columns with a **datetime** specification. The format is as per Java SimpleDateFormat format.
showColumnFilters	Boolean	true	Determines if column filters will be displayed in the output.
dtOptions	Object		Options related to the DataTables output. Available options are as per the DataTables documentation. Where a `<th>` column class name is required, you can use `rcol-<column name>`. Spaces and other characters apart from numbers, letters, underscores and hyphens will be replaced with a hyphen; so for a report column named `Due Date`, the class name to use would be `rcol-Due-Date`.

Another alternative for querying MongoDB is to use the Calcite MongoDB JDBC driver under the JDBC datasource type. This would allow use of normal SQL to query MongoDB. Yet another alternative for querying MongoDB is to install and use Apache Drill. Drill allows you to use normal SQL to query a MongoDB database. When using drill, you wouldn't use the MongoDB datasource type or the MongoDB report type. You would use the JDBC datasource type with the drill driver, and then use any of the other report types e.g. Tabular etc. The drill jdbc driver can be obtained from the drill installation and copied to the ART_HOME\WEB-INF\lib directory.

Note:

- Instead of using the MongoDB report type, one can specify any of the other report types e.g. Tabular, enable the **Use Groovy** option and use the kind of code as given above to generate the report.

12.2.67. View

Used as a basis for creating self service reports. The view report will define the columns that should be available for selection and the joins and basic conditions for the reports. The report source will contain placeholders to be replaced when the view or self service reports are run. Examples are given below.

Example with a MySQL datasource

```
select #columns#
from my_table
where #condition#
#limitClause#
```

Example with an SQL Server datasource

```
select #limitClause# #columns#
from my_table
where #condition#
```

Example with the use of a join

```
select #columns#
from my_table a
inner join my_other_table b
on a.id=b.id
where #condition#
#limitClause#
```

The **#columns#** placeholder will be replaced with the selected columns. The **#condition#** placeholder will be replaced by the selected report conditions. The **#limitClause#** placeholder will be replaced by the number of records to display when previewing the view or self service report and needs to be placed in the appropriate place according to the database in use.

A number of options can be configured in the **Options** field. This would be specified under a property named **view** in the JSON definition.

Property	Data Type	Default	Description
columns	String	*	The specification of the columns to be made available for selection e.g. `col1, col2`
omitColumns	String[]		Column names that should be omitted from the list of available columns

Property	Data Type	Default	Description
columnLabels	Object[]		Specification of column labels. Each object specifies a column name as the property and the column label as the value.
columnDescriptions	Object[]		Specification of column descriptions. Each object specifies a column name as the property and the column description as the value. The column description is displayed as a tooltip when selecting columns.
conditionColumns	String[]		Column names that should be included in the list of condition columns. If not specified, all available columns will be included in the conditions list.
omitConditionColumns	String[]		Column names that should be omitted from the list of condition columns
sortColumns	Boolean	true	Whether the available columns and condition columns should be sorted
valueSeparator	String	,	The character used to separate multiple values in conditions where multiple values may be specified e.g. when using the **in** operator.
filterOptions	Object`[]`		Specification of filter options for particular condition columns as per the jQuery QueryBuilder Filters documentation. Each object would have a **column** property which would specify the name of the column and an **options** property which would be an object as per the jQuery QueryBuilder documentation e.g. `"filterOptions":[{"column": "myid", "options": {"input": "number"}}]`. You can use the column name **all** to specify options that should be applied to all filters.
limitClause	String		The syntax of the limit clause e.g. `limit {0}`. The placeholer "{0}" will be replaced with the configured limit value. This option can be specified on the datasource so that it doesn't need to be repeated for every view. It is also not necessary to specify it for common databases.

Property	Data Type	Default	Description
limit	Integer		The number of records to return when testing/running the view. The view report is not intended to be used as an actual report and so if this option is not defined, the view report will return a maximum of 10 records when you test/run it. This limit is not applied to the self service reports that will be created, but is only applicable for the view report.

12.2.68. File

Outputs the contents of a query as it is e.g if using the "FOR XML" clause in SQL Server. A number of options can be specified in the **Options** field.

Property	Data Type	Default	Description
extension	String	txt	The extension to use for generated file names

12.2.69. Link

Opens a link as provided in the **Link** field.

Note:

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

12.3. General Report Options

A number of options can be specified in the **Options** field of a report that are applicable to several report types. These options are defined in JSON format with the following possibilities.

12.3.1. General Options

Property	Data Type	Description
i18n	Object	Allows for localization of some aspects of a report. It can have the properties **name**, **shortDescription**, **description**, with each property being an array of objects providing the locale and the appropriate localized string. e.g. `"i18n": { "name": [{"en, sw": "A report name"}, {"pt": "Another report name"}], "description": [{"lt": "Sample help text"}] }`. In addition, the **i18n** object can have a property named **columnNames** that enables localization of data output column names. This property also holds an array of objects defining the localized column names e.g. `"i18n": { "columnNames": [ {"col1": [{"lt": "Some name"}]}, {"2": [{"lt": "A name"}]} ]}`. You can specify a column name e.g. "col1" or a column index e.g. "2", the first column being index 1.
c3	Object	Allows for custom configuration of C3 Chart output with tabular reports. The properties are same as those specified for the **C3.js** report type.
plotly	Object	Allows for custom configuration of Plotly Chart output with tabular reports. The properties are same as those specified for the **Plotly.js** report type.
refreshPeriodSeconds	Integer	Enables a report to be re-run or refreshed automatically after the first run. Minimum is 5 seconds.
queryTimeoutSeconds	Integer	Enables setting of a query timeout for a report that uses a JDBC query. If the timeout expires before the query completes, the query execution will stop and an exception will be thrown. The value if defined should be >= 0, with 0 meaning no timeout.
fileName	String	Allows for specifying the base file name that should be used for report output. Parameter values can be included by specifying the parameter name enclosed with # e.g. `#param1#`. To include the default generated name, use the tag `{default}` e.g. `#param1#{default}`. This base file name does not include the extension. The extension will be automatically provided depending on the report and report format.

12.3.2. PDF Options

For report types that can generate pdf output, the following options can be specified.

Property	Data Type	Default	Description
pdfCanPrint	Boolean	true	Whether the Print menu is enabled in the generated pdf
pdfCanCopyContent	Boolean	true	Whether one can select and copy text in the generated pdf
keyLength	Integer	128	The encryption key length to be used. Can only be either 40, 128 or 256.
preferAes	Boolean	true	Whether the AES algorithm should be used if available

12.3.3. Clone Options

For clone reports (reports where the **Source Report** field points to some parent report ID), the following options can be specified.

Property	Data Type	Default	Description
useParentParameters	Boolean	true	Whether the clone report will use parent report's parameters, or if false, use its own parameters.
useParentRules	Boolean	true	Whether the clone report will use the parent report's rules, or if false, use its own rules.

12.3.4. Tabular Options

For tabular report types, the following options can be specified.

Property	Data Type	Description
imageColumns	String[]	An indication of the column indices or column names for blob columns that contain images. The first column has an index of 1. e.g. `["2"]` or `["image"]` or `["3", "image"]`. A note that ods and odt report formats will not display images.
dtOptions	Object	Options to be used with the **htmlDataTable** report format. Available options are as per the DataTables documentation. Where a `<th>` column class name is required, you can use `rcol-<column name>`. Spaces and other characters apart from numbers, letters, underscores and hyphens will be replaced with a hyphen; so for a report column named `Due Date`, the class name to use would be `rcol-Due-Date`.
dtExtraOptions	Object	Extra options for the **htmlDataTable** report format

12.3.5. Html Data Table Report Format Extra Options

For Tabular reports, an option named **dtExtraOptions** can be specfied to provide some extra configuration of how data table output will be displayed. This option can also be specified in the **Settings** page, under the **Options** field to give a default to be used for all tabular reports. The object has the following properties.

Property	**Data Type**	**Default**	**Description**
pdf	Boolean	false	Whether pdf export will be available in the output

12.3.6. Template Result Options

For jxls, xdocreport, freemarker, velocity and thymeleaf report types, the following options can be specified to determine the column names to use with the **results** object.

Property	**Data Type**	**Default**	**Description**
useColumnLabels	Boolean	true	Whether to use column labels (aliases) as the property names. If false, column names will be used.

12.4. Running a report via URL

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

If direct URL access is needed without requiring the user to log in to ART beforehand, **public=true** must be included in the URL together with a **user** parameter indicating the username under whose permissions the report will be run. In addition, this user must have the **Public User** field set to **Yes**. e.g. `http://localhost:8080/art/runReport?reportId=1&public=true&user=test`

12.4.1. Report Options

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

Option	**Data Type**	**Comments**
p-{parameter_name}	String	Specifies report parameters e.g. `p-duedate=2010-05-06`
reportFormat	String	Specifies the report format that should be used for the output generated by the report e.g. `reportFormat=pdf`. Not all report formats are available for all report types.

Option	Data Type	Comments
splitColumn	Integer	For group reports, the group column i.e. the column index by which grouping should be done e.g. `split-Column=2`
public	Boolean	If present, determines that the user doesn’t need to be logged in to ART to view the report results e.g. `public=true`. If authentication is required, don’t include this parameter.
showSelectedParameters	Boolean	If present, indicates that the used or selected report parameters should be included in the report output e.g. `showSelectedParameters=true`. Omit the parameter entirely if this is not required.
showSql	Boolean	If present, indicates that the final SQL used to generate the report results is shown in the browser e.g. `showSql=true`. Omit the parameter entirely if this is not required.
allowSelectParameters	Boolean	If present, indicates that the parameters box should be displayed in the browser to allow the user to re-run the report by changing the reports parameter values.
startSelectParametersHidden	Boolean	If present, indicates that the parameters box should start as hidden/collapsed. The parameters box can then be displayed by clicking on the **Parameters** button.
prettyPrint	Boolean	For **json** and **jsonBrowser** report formats, determines if the json output should be pretty printed
refreshPeriodSeconds	Integer	Enables a report to be re-run or refreshed automatically after the first run. Minimum is 5 seconds.
showColumnFilters	Boolean	For the **htmlDataTable** report format, if present indicates that column filters should be included in the output
isFragment	Boolean	If set to true e.g. `isFragment=true`, the report information header and footer showing the start time of execution and the number of rows retrieved will not be shown.
directDownload	Boolean	For file output, determines if the file should be served directly instead of showing a "Download" link

12.4.2. Chart Options

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

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

12.4.3. Examples

Example 1: (single-value parameters)

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


runs report 120 in html format.
The #startdate# parameter (p-startdate) is set to 2006-04-04
and the #description# parameter (p-description) is set to "Desktop"
```

Example 2: (chart)

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

Example 3: (group)

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


runs report 122 in group on 2 columns mode (splitColumn parameter).
The parameter named param is set to "abc".
```

Example 4: (pdf, public)

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


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

Example 5: (dashboard)

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


runs report 124, setting values for parameters param1 and param2
```

Example 6: (show selected parameters)

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

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

Example 7: (multi-value parameters)

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

the #category# multi-value parameter (p-category) has 3 values defined:
Laptops, Desktops, Tablets
```

Example 8: (default parameter values)

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

generates report using default parameter values
```

Note:

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

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

12.5. Report Formats

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

Report Format	Display Name	Description
html	Browser	Shows results in a web page
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.
htmlFancy	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 (xls format). Excel/OpenOffice/LibreOffice compatible
xlsZip	Spreadsheet (zip xls)	Creates a Microsoft Excel spreadsheet file (xls format), compressed using Zip
xlsx	Spreadsheet (xlsx)	Creates a Microsoft Excel spreadsheet file (xlsx format)
slk	Spreadsheet (slk)	Creates a Microsoft Excel spreadsheet file (slk format)
slkZip	Spreadsheet (zip slk)	Creates a Microsoft Excel spreadsheet file (slk format), compressed using Zip
tsv	File (tsv)	Creates a tab-separated-values file
tsvZip	File (zip tsv)	Creates a tab-separated-values file, compressed using Zip

Report Format	Display Name	Description
tsvGz	File (gzip tsv)	Creates a tab-separated-values file, compressed using GZip
pdf	Document (pdf)	Creates a pdf file
png	Image (png)	Creates a png file. Only available for charts.
xml		Creates xml output displayed on the browser
rss20		Output data as a rss2.0 compliant feed
docx	Document (docx)	Creates a Microsoft Word document file (docx format)
odt	Document (odt)	Creates an Open Document Format text document file (odt format)
ods	Spreadsheet (ods)	Creates an Open Document Format spreadsheet file (ods format)
pptx	Document (pptx)	Creates a Microsoft PowerPoint document file (pptx format). Only available for XDocReport - PPTX reports.
json	JSON	Outputs data in json format
jsonBrowser	JSON (Browser)	Outputs data in json format, for display in a browser. Data is output in a `<pre>` tag.
csv	File (csv)	Creates a comma-separated-values file. Can also use another delimiter as defined in the report **Options** field - using the **delimiter** property.
csvZip	File (zip csv)	Creates a comma-separated-values file, compressed using Zip
txt	File (txt)	For **Fixed Width** reports, creates a fixed-width text file.
txtZip	File (zip txt)	For **Fixed Width** reports, creates a fixed-width text file, compressed using Zip
pivotTableJs	Browser (Pivot Table)	For tabular reports, shows the results as a dynamic pivot table
c3	Browser (C3 Chart)	For tabular reports, shows the data as a c3 chart
plotly	Browser (Plotly Chart)	For tabular reports, shows the data as a plotly chart
file	File	For **File**, **FreeMarker**, **Velocity** and **Thymeleaf** reports, outputs the results to a file
fileZip	File (zip)	For **File**, **FreeMarker**, **Velocity** and **Thymeleaf** reports, outputs the results to a file, compressed using Zip

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

Note:

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

13. Parameters

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

13.1. Creating a new parameter

You can create a parameter either by using the **Configure | Parameters** menu and selecting the **Add** button or from the **Reports Configuration** page, finding the report you want to add a parameter to, selecting the **More | Parameters** option and selecting the **Add New** button.

Field	Description
ID	An auto-generated ID used to identify the parameter
Name	The name of the parameter as it appears in reports' SQL source. It should not contain spaces or other special characters e.g. `!"#$%&'()*+,./:;<=>?@[\]^{}~`.
Description	A description for the parameter
Parameter Type	Whether the parameter will have a single value (**Single-Value**) or whether it can have multiple values (**Multi-Value**)
Label	The label for the parameter in the select parameters page
Help Text	Help text displayed as a tooltip in the select parameters page
Placeholder Text	Placeholder text displayed in the field in the select parameters page
Data Type	The type of data that the parameter can hold. A **File** data type will return a list of objects of type CommonsMultipartFile that would be available from the groovy source of the report.
Date Format	For **Date** and **DateTime** data types, the date format to use. Leave blank to use the default. The format is as per Java SimpleDateFormat format.
Multiple Files	For the **File** data type, whether the user is allowed to select multiple files
Accept	For the **File** data type, a comma separated list of file extensions that will be displayed by the file input control e.g. `.csv, .txt`. This goes to the accept attribute of the file input.
Default Value	The value for the parameter to use when the parameter is first displayed in the select parameters page, or the value to use if the report is run by direct url and the parameter is not specified within the url. For **Multi-Value** parameters, multiple values can be specified, with each value on a new line.
Default Value Report	A report that provides the default values. This needs to be an LOV report.

Field	Description
Use Default Value In Jobs	Whether the default value or values should be used as the parameter value when running jobs
Template	An optional javascript file for additional configuration. For **DateRange** data types, the file can be used to customize the date range picker. You can override the **overallOptions**, **ranges** or **localeOptions** variables as per the **WEB-INF\jsp\daterangeInput.jsp** file. For masks, the file may contain additional configuration options. For "mask1" masks, specify the additional options in a variable named **mask1Options**.
Shared	Whether the parameter is available for selecting when defining report parameters, through the **Add Existing** option.
Hidden	Whether the parameter is displayed in the select parameters page
Use LOV	Whether the parameter uses a dropdown to provide a list of values to choose from in the select parameters page
LOV Report	If **Use LOV** is specified, the report that generates the available values. This needs to be an LOV report.
Drilldown Column Index	The index of the column that will be set with the parameter's value when used with drilldown reports. The first column has an index of 1.
Fixed Value	Whether the parameter uses fixed values
Allow Null	Whether a checkbox will be displayed for the user to specify passing NULL as a parameter value. If null values are passed for a parameter, ensure that you use the x-parameter syntax otherwise you will likely get incorrect results. To specify NULL value via url, add `-null` to the parameter name and include that in the url e.g. `&p-param1-null=true&p-param2=test` would pass a NULL value for the parameter named `param1`.
Options	Options used to further configure the parameter. The options are specified in JSON format.

13.2. Parameter in SQL Source

13.2.1. Parameterized value

To pass parameter values to an SQL query, in the report's sql source, surround the parameter name with the # character i.e. `#param_name#`. This will replace that parameter placeholder occurrence with `?` in the SQL query and the database will put the parameter value as appropriate when the query is run.

13.2.2. Direct substitution

If you would like the parameter value to be directly applied to the SQL source before the query is run, use the syntax `!#param_name#`. This therefore means that the sql query will be prone to sql injection as opposed to using the parameterized syntax. For multi-value parameters, the parameter values will be separated by a comma.

13.3. Single-Value Parameters

Single-value parameters are used to pass single values to the SQL query. For example, the following query has two parameters.

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

Take the following steps to use a single-value parameter

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

Note:

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

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

- **Workaround for boolean parameters**

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

Example main query

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

Example static LOV query for boolean values

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

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

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

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

13.4. Multi-Value Parameters

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

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

Take the following steps to use a multi-value parameter

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

Note:

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

13.5. X-Parameter Definitions

There exists an alternative syntax for specifying `IN`, `NOT IN`, `=` and `<>` comparisons. This syntax takes the form `$x{<comparator>,<column_name>,<parameter_name>}`. It should be noted that with x-parameter definitions, the parameter name is case sensitive and must be specified exactly the way it is defined in the parameter configuration.

13.5.1. IN, NOT IN

This is used with multi-value parameters. IN clauses take the format `$x{in,<column_name>,<parameter_name>}` and NOT IN clauses take the format `$x{notin,<column_name>,<parameter_name>}` e.g.

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

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

13.5.2. EQUAL, NOT EQUAL

This is used for single-value parameters. Equal comparators take the format `$x{equal,<column_name>,<parameter_name>}` and would typically be equivalent to `column_name=#parameter_name#`. In case NULL is passed, the syntax would be converted to `column_name IS NULL`. Not equal comparators take the format `$x{note-qual,<column_name>,<parameter_name>}` and would typically be equivalent to `column_name<>#parameter_name#`. In case NULL is passed, the syntax would be converted to `column_name IS NOT NULL`.

13.6. Parameter Options

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

13.6.1. i18n Options

Certain parameter fields can be localized by specifying an **i18n** object in the Options field with the following possible values.

Property	Data Type	Description
label	Object[]	Localization for the parameter label. Each object has the language code as the key and the translation as the value. Multiple language codes can be specified separated by comma e.g. `{"label": [{"en, sw": "Test Label"}, {"pt": "Another Label"}]`
helpText	Object[]	Localization for the parameter help text. Similar format to label.
defaultValue	Object[]	Localization for the parameter default value. Similar format to label.
placeholderText	Object[]	Localization for the parameter placeholder text. Similar format to label.

A sample **i18n** configuration may be `"i18n": { "label": [{"en, sw": "Test Label"}, {"pt": "Another Label"}], "helpText": [{"lt": "Sample help text"}] }`

13.6.2. Date Range Options

For **DateRange** parameters, options can be specified in a **dateRange** object with the properties below. ART uses the Date Range Picker library to provide the date range picker. Some of the options are given below. Others are available as per the Date Range Picker documentation.

Property	Data Type	Default Value	Description
fromParameter	Object		A specification of the parameter that should get the "From" date of the date range. This object has two properties, **name** which specifies the parameter name and **format** which specifies the date format of the parameter. The format is as per Java SimpleDateFormat format and the default is **yyyy-MM-dd**.
toParameter	Object		A specification of the parameter that should get the "To" date of the date range. Similar specification to **fromParameter**.
format	String	yyyy-MM-dd	The format of dates displayed in the date range picker text input. The format is as per Java SimpleDateFormat format. If you use the **locale.format** property, that will need to be in Moment.js format.
ranges	String[]	`["default"]`	Pre-defined, custom date ranges available for selection. Can be set to **null** if custom ranges are not required. Possible values include **today**, **yesterday**, **last7Days**, **last30Days**, **thisMonth**, **lastMonth**, **thisQuarter**, **lastQuarter**, **thisYear**, **lastYear**, **thisWeek**, **lastWeek**, **yearToDate**, **monthToDate**, **quarterToDate**, **weekToDate**.
startDate	String		A specification of the start date when the date range picker is first displayed. You can use syntax like "add days -1", "add months 2" etc to specify a date in the past or future. In this case the resulting date is as per the **format** option.
endDate	String		A specification of the end date when the date range picker is first displayed. Similar to **startDate**.

13.6.3. Other Options

Additional properties can be specified in the Options JSON as follows

Property	Data Type	Description
mask1	Object	Enables use of a mask for the parameter as per the RobinHerbots Inputmask e.g. `"mask1": { "mask": "99-aa", "autoUnmask": true }`

14. Expressions

Expressions enable including of dynamic content in certain fields. Fields that can have expressions include

- Job fixed file name
- Job email subject
- Job email message
- Parameter values and default values
- Report source

14.1. Field Expressions

Field expressions substitue certain known items. Field expressions start with `f[` and end with `]f`. Field expression names are case sensitive.

14.1.1. Username

You can use a field expression to substitute the username of the currently logged in user, or for jobs, the job owner. Use the value `f[username]f` to substitute the username.

14.1.2. Dates

You can substitute the current date by using the expression `f[date]f`. This will output the current date in **yyyy-MM-dd** format e.g. 2017-11-10. You can use the expression `f[datetime]f` to substitute the current date and time in **yyyy-MM-dd HH:mm:ss.SSS** format e.g. 2017-11-10 11:25:21.045. Both date and datetime field expressions can also specify a different date other that "now" and additionaly specify the output format, output format locale, input format and input format locale. These additional items are specified separated by `|` i.e `f[date field name|date specification|output format|output format locale|input format|input format locale]f`.

The date specification component can be the string "**today**" to specify today’s date with the time component as zero, it can be the string "**now**" to specify the current date and time, or a date string in certain numeric formats e.g. `yyyy-MM-dd` or `yyyy-MM-dd HH:mm:ss`. In addition one can have simple date arithmetic by having a string starting with "**add**" and then specifying an offset from the current date e.g. `add days 1` or `add months -1`. The duration offsets that can be specified include **days**, **weeks**, **months**, **years**, **hours**, **minutes**, **seconds**, **milliseconds**.

The output format component can be used to specify the desired output format of the resulting date using Java SimpleDateFormat syntax. Additionaly, a locale can be specified to define the locale which should be used with the given output format.

If the date specification is not in one of the common numeric formats, the format it is in can be specified in the input format section, and if required, the input format locale can be specified.

14.2. Groovy Expressions

Groovy code can be used to provide dynamic text. Groovy expressions start with `g[` and end with `]g` e.g. `g[1+1]g`. You can have longer code which includes imports etc. Groovy code is by default passed through a sandbox so if a **SecurityException** is thrown, the relevant class needs to be added to the **WEB-INF\groovy-whitelist.txt** file, or alternatively, turn off the sandbox in the **WEB-INF\art-custom-settings.json** file.

15. Dynamic SQL

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

15.1. Using Groovy

You can use groovy in the SQL source section to achieve dynamic sql. Set the **Use Groovy** field of the report to specify that you are using groovy in the sql source.

The groovy script may return a string that has the sql to be executed or it may return a `java.util.List` of `Map<String, Object>` or GroovyRowResult objects that contain the data to be output. If returning a List of objects, the following options can be specified in the **Options** field of the report.

Property	Data Type	Description
columns	String[]	Names of columns that should be included in the output. This is optional and may be useful to specify the order of columns to be displayed.
columnDataTypes	Object[]	Each object specifies a column name and the data type of that column e.g. `[{"col1": "numeric"}]`. The possible data types are **string**, **numeric**, **date** and **datetime**. The default is `string`.
columnLabels	Object[]	Each object specifies a column name and the heading to be used for that column e.g. `[{"col1": "Column 1"}]`. If not specified, the column name is used as the heading.

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

15.1.1. Accessing parameter values

To access parameter values in groovy code, use the name of the parameter and call the **value** method e.g. `param1.value`. For **File** parameters, use the parameter name directly instead of calling the value method. Also, file parameter values are always passed as a list so access the parameter variable using list syntax e.g. `fileParam1[0]`. If the file parameter has **Multiple Files** enabled, this list may have more than one value.

15.1.2. Examples

Integer parameter

```
def sql

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

File parameter

```
//assumes the report has a file parameter named importFile
if (importFile == null) {
    println(’report run from url or file parameter not available’)
} else if (importFile[0].size == 0) {
    println(’empty file or no file selected’)
} else {
    InputStream inputStream = importFile[0].getInputStream();
    Scanner sc = null;

    try {
    sc = new Scanner(inputStream, "UTF-8");
        while (sc.hasNextLine()) {
            String line = sc.nextLine();
            println(line);
        }
    } finally {
        if (inputStream != null) {
            inputStream.close();
        }
        if (sc != null) {
            sc.close();
        }
    }
}
```

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

15.2. Using XML tags

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

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

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

For example in the following query:

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

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

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

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

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

15.2.1. 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)

15.2.2. Examples

Dynamic OR/AND selection:

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

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

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

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

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

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

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

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

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

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

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

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

16. Multiple Statements

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

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

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

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

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

17. Dynamic Datasources

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

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

When you run the report, it will use the value of the datasource parameter to determine which data-source to run the report on. The value of this parameter should be a datasource id or datasource name. If it passes the "isNumeric" test, it will be assumed that it is referring to a datasource id, otherwise a check against datasource names will be done.

18. Rules

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

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

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

- **Link the rule to a report**
  On the report definition page, select the **Uses Rules** option
  In the SQL source section, use the special, hard coded placeholder **#rules#** where you want the rule values to go e.g

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

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

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

Example:

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

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

Note:

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

19. Chained Parameters

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

19.1. Example

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

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

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

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

- Create a Dynamic LOV report named **Cities** that will display city names. The query needs to have a parameter label for the country parameter. This will be replaced at runtime with the country selected.

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

- Once you have saved the Cities report, click on the **Parameters** option to assign the countryId parameter

- Use the **Add New** option to create a Single-Value parameter with a Name of **countryId**, Label of **Country**, Data Type of **Integer**, set it to **Shared**, set it to **Use LOV** and select **Countries** as the **LOV Report**.

- Create the main report. The SQL for the report will need to have a placeholder for the cityId parameter.

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

- Once you have saved the main report, click on the **Parameters** option to allocate the report parameters

- Use the **Add Existing** option to add the **countryId** shared parameter created earlier
- Use the **Add New** option to create a Single-Value parameter with a Name of **cityId**, Label of

City and Data Type of **Integer**. Once you have saved this parameter, click on its **Edit** button on the Report Parameters page and set the **Chained Parents** field to **countryId** i.e. the name of the parent parameter for this chained parameter. If the parent parameter data type is **Date** or **DateTime**, in the **Chained Parents** field, put the name of the parent parameter and a "-hidden" suffix e.g. **countryId-hidden**, and then set the **Chained Depends** field to the name of the parent parameter e.g. **countryId**.

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

20. Encryptors

Encryptors allow you to encrypt report output. Use the **Configure | Encryptors** menu to manage encryptors.

Field	Description
ID	Auto-generated ID used to identify the encryptor
Name	A name to identify the encryptor
Description	A description for the encryptor
Active	Defines whether the encryptor will be applied to reports
Encryptor Type	Specifies the type of encryption to be used. The **AES Crypt** type will produce files that can be decrypted using the AES Crypt program. The AES Crypt encryptor uses 256-bit keys therefore your java version must support 256-bit encryption. For OpenJDK, this support is available by default. For Oracle Java, JCE Unlimited Strength Jurisdiction Policy files must be available in your JVM. These files are available and enabled by default from Oracle Java 8 u162. The **OpenPGP** type will produce files that can be decrypted using any OpenPGP-compliant program. **Password** encryptors are used to specify open and modify passwords for xlsx and pdf output.
AES Crypt Password	For AES Crypt encryptors, the password that will be used to encrypt the file. You cannot use a blank password.
OpenPGP Public Key File	For OpenPGP encryptors, the public key file to use to encrypt the file. The key file can either be in binary or ascii armored format.
OpenPGP Public Key String	For OpenPGP encryptors, the ascii armored public key text. You would set either the public key string or the public key file. If you set both, the public key string will be used.
OpenPGP Signing Key File	For OpenPGP encryptors, the key file to use to sign encrypted files. This file is optional, if you do not wish for files to be signed. The key file can either be in binary or ascii armored format.
OpenPGP Signing Key Passphrase	For OpenPGP encryptors, if a signing key is used, the passphrase for that key (not blank).
Open Password	For Password encryptors, the file open password to be used with xlsx, pdf, docx, odt, ods output
Modify Password	For Password encryptors, the file edit password to be used with xlsx and pdf output

21. LOV Reports

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

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

Users see values from the `location_name` column while the actual parameter match is performed using values from the `location_id` column.
If the parameter options don't come from a database, you can use a static lov with something like the following in the sql source. If the parameter value and the display value are different, separate them with a |

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

21.1. Example

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

In order to proceed with this example you need to:

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

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

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

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

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

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

Select the ART database as the Datasource and Save.

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

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

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

Note:

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

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

22. Drill Down Reports

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

The main report can be a tabular report displayed in one of the html report formats, or a chart. The drill down report can be of any type.

22.1. Defining the main report

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

22.2. Defining the drill down report

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

22.2.1. Example

- **Sales Summary** (Main report)

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

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

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

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

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

On the Main report, select the **More | Drilldowns** option and select **Add** to create the Drilldown definition.

When running the Main Report, a new column will appear on the right. Click on it and the drill down report will be executed - the `#region#` parameter will match with the first column value.

22.2.2. Drilldown definition

Field	Description
ID	An auto-generated ID used to identify the drilldown
Drilldown Report	The drill down report
Header Text	The column title/header of the drilldown link column
Link Text	The text of the drilldown link
Report Format	The report format that will be used to display the drill down report results
Open In New Window	Whether the result of the drill down report should be opened in a new window
Allow Select Parameters	Whether the user will be provided with the parameters box to select/change parameters after the drill down report has run if they wish to re-run the report
Run Immediately	Whether the drill down report will run immediately

22.3. Using the main report's parameters

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

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

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

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

So the Drill Down Report would look something like

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

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

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

Note:

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

23. RSS Feeds

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

This is achieved in three steps:

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

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

Element Name	Description
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>[Report Name]</title>
 <link>[As configured in ART settings]</link>
 <description>[Report Name]</description>
 <pubDate>[Current date]</pubDate>
 <generator>http://art.sourceforge.net</generator>
 <item>
  <title>[resultset "title" column]</title>
  <description>[resultset "description" column]</description>
  <guid>[resultset "guid" column]</guid>
  <pubDate>[resultset "pubDate" column]</pubDate>
 </item>
 <item>
  ...
 </item>
 ...
</channel>
</rss>
```

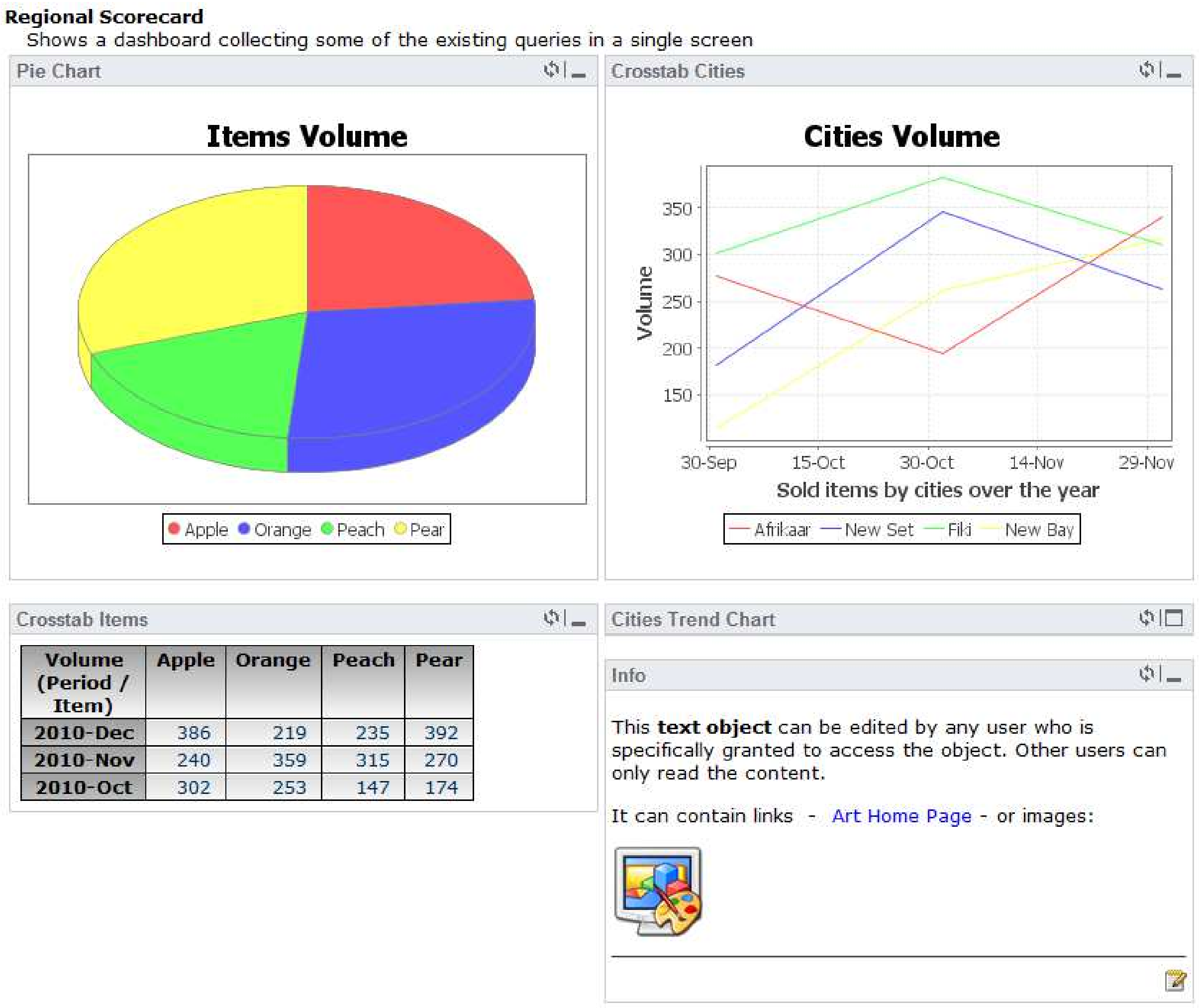
Note:

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

24. Dashboards

Dashboards are used to display multiple reports on a single page. Most report types can be included in a dashboard with exceptions including Saiku, JPivot and Link reports.

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



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

Volume (Period / Item)	Apple	Orange	Peach	Pear
2010-Dec	386	219	235	392
2010-Nov	240	359	315	270
2010-Oct	302	253	147	174

This **text object** can be edited by any user who is specifically granted to access the object. Other users can only read the content.

It can contain links - Art Home Page - or images:

24.1. Regular Dashboards

You define a dashboard by creating a report of type **Dashboard** and specifying the details of the dash-board using XML syntax in the **Source** field.

24.1.1. Dashboard Properties

A number of tags are used to define properties of the dashboard.

Tag	Data Type	Description
`<DASHBOARD>`	None	The parent/containter tag for the dashboard definition
`<COLUMN>`	None	The parent/container tag for a dashboard column. A dashboard can have any number of columns.

24.1.2. Column Properties

A dashboard can have a number of columns, enclosed within `<COLUMN>` tags. These columns have a number of properties which are specifed using the following tags.

Tag	Data Type	Default	Description
`<SIZE>`	String	auto	The size of the column. Either **small**, **medium**, **large** or **auto**.
`<PORTLET>`	None		The parent/containter tag for a portlet. A column can have any number of portlets.

24.1.3. Portlet Properties

A dashboard column can have a number of portlets, enclosed within `<PORTLET>` tags. These portlets have a number of properties which are specifed using the following tags.

Tag	Data Type	Default	Description
`<TITLE>`	String		The title of the portlet
`<REPORTID>`	String		A report ID of an ART report that should be displayed in the portlet e.g. `25`. You can add parameters after the report ID, being careful to use `&amp;` where & would be used in a url e.g. `25&amp;reportFormat=htmlGrid`
`<URL>`	String		An external url that should be displayed in the portlet e.g. `http://www.example.com`. If both url and report id tags are present, the content of the report id tag will be used.
`<REFRESH>`	Integer	-1	The time in seconds after which the portlet refreshes itself. -1 means no auto-refresh. The minimum refresh time allowed is 5 seconds. Setting a low value may overload your servers/databases. A portlet can be refreshed manually by clicking on the refresh button in the portlet header.
`<ONLOAD>`	Boolean	true	Whether the report or url content is retrieved when the dashboard is run. If set to false, the portlet content is not loaded initially and can be loaded later manually by using the refresh button.

24.1.4. Example Syntax

```
<DASHBOARD>

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

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

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

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

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

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

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

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

</DASHBOARD>
```

Note:

- Tag names are case sensitive (use uppercase)
- If you need to use xml special characters within tag content, ensure to replace them with the relevant xml character entities e.g. if the `<URL>` tag should contain a url like `http://www.example.com/page.html?param1=1&param2=test`, you'll need to replace the & character with `&amp;`, to have the url as `http://www.example.com/page.html?param1=1&amp;param2=test`. For a list of the 5 special xml characters and their respective replacement strings or character entities, see http://support.esri.com/technical-article/000005870
- For pdf output of dashboards, only tabular and standard chart reports will be included in the pdf output.

24.2. Gridstack Dashboards

Gridstack dashboards allow for specifying exact positioning of report items within a dashboard, specifying the height and width that report items should occupy, and allow for resizing and moving of items using drag-and-drop. The gridstack.js library is used to provide this functionality. Use the **Dashboard: Gridstack** report type to create a gridstack dashboard.

24.2.1. Dashboard Properties

Gridstack dashboards are also defined using xml syntax. A number of tags are used to define properties of the dashboard.

Tag	Data Type	Default	Description
`<DASHBOARD>`	None		The parent/containter tag for the dashboard definition
`<DASHBOARDWIDTH>`	Integer	12	The number of columns to be taken up by the grid, forming the basis of the horizontal axis (x axis) of the grid co-ordinates.
`<FLOAT>`	Boolean	false	Whether report items can be moved to arbitrary locations on the dashboard. If false, grid items can only snap to given locations.
`<ANIMATE>`	Boolean	false	Whether animation is used when grid items are moved around
`<DISABLEDRAG>`	Boolean	false	Whether dragging of items is disabled
`<DISABLERESIZE>`	Boolean	false	Whether resizing of items is disabled
`<CELLHEIGHT>`	String	60px	The dimensions of one unit of the grid co-ordinates on the vertical axis (y axis). It can be defined as **px**, **em** or **rem** units e.g. 10em or 10rem. It can also be defined as "**auto**", in which case the height will be calculated from the cell width.
`<VERTICALMARGIN>`	String	20px	The size of the vertical gap between cells. It can be defined as **px**, **em** or **rem** units e.g. 2em or 2rem.
`<ALWAYSSHOWRESIZEHANDLE>`	Boolean	false	Whether the resizing handles on the bottom corners of a cell are shown, even if the user is not hovering over the item.
`<DASHBOARDHEIGHT>`	Integer	0	The maximum number of rows or y units on the grid co-ordinate system. Items cannot be resized or moved beyond this limit. A value of 0 means no maximum.
`<ITEM>`	None		The parent/container tag for a dashboard item. The dashboard can have any number of items.

24.2.2. Item Properties

A gridstack dashboard can have a number of items, enclosed within `<ITEM>` tags. These items have a number of properties which are specifed using the following tags.

Tag	Data Type	Default	Description
`<XPOSITION>`	Integer		The x axis position of the item. From 0 to `<DASHBOARDWIDTH>`
`<YPOSITION>`	Integer		The y axis position of the item
`<WIDTH>`	Integer	2	The width of the item. This defines the number of columns that the item will occupy, in relation to the number of columns of the whole grid as defined by the dashboard's `<DASHBOARDWIDTH>` property.
`<HEIGHT>`	Integer	2	The height of the item. This defines the number of rows that the item will occupy, one row being the size defined in the dashboard's `<CELLHEIGHT>` property.
`<AUTOWIDTH>`	Boolean	false	Whether the width should be automatically set according to the item contents
`<AUTOHEIGHT>`	Boolean	false	Whether the height should be automatically set according to the item contents
`<NORESIZE>`	Boolean	false	Whether element resizing is disabled
`<NOMOVE>`	Boolean	false	Whether element moving is disabled
`<AUTOPOSITION>`	Boolean	false	Whether to ignore the `<XPOSITION>` and `<YPOSITION>` and instead place the element in the first available position.
`<LOCKED>`	Boolean	false	Whether the widget will be locked. If it is locked, another widget will not be able to move it during dragging or resizing. Even though it is locked, the widget can still be dragged or resized. You need to specify the `<NORESIZE>` and `<NOMOVE>` options to prevent this and thereby completely lock the widget.
`<MINWIDTH>`	Integer	0	The minimum width beyond which the item cannot be resized
`<MINHEIGHT>`	Integer	0	The minimum height beyond which the item cannot be resized
`<MAXWIDTH>`	Integer	0	The maximum width beyond which the item cannot be resized. A value of 0 means no maximum.
`<MAXHEIGHT>`	Integer	0	The maximum height beyond which the item cannot be resized. A value of 0 means no maximum.

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

24.2.3. Example Syntax

```
<DASHBOARD>

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

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

</DASHBOARD>
```

24.3. Tabbed Dashboards

Both regular and gridstack dashboards can have their components displayed within tabs. This is done by adding the optional `<TABLIST>` tag to the dashboard xml and specifying the number of tabs, and which portlets/items should appear in which tabs.

24.3.1. Tablist Properties

The `<TABLIST>` tag is used to specify that the dashboard portlets/items should be displayed in tabs. A tablist has a number of properties which are specifed using the following tags.

Tag	Data Type	Default	Description
`<DEFAULTTAB>`	Integer	1	The index of the tab that will be active when the dashboard is displayed. The first tab has an index of 1.
`<TAB>`	None		The parent/container tag for a tab. A tablist can have any number of tabs.

24.3.2. Tab Properties

A tablist can have a number of tabs, enclosed within `<TAB>` tags. The tabs have a number of properties which are specifed using the following tags.

Tag	Data Type	Default	Description
`<TITLE>`	String		The title/heading of the tab. Even though this tag is optional, it is recommended to set it so as to have better identification of the different tabs.
`<ITEM>`	Integer		The index of the item to display, as it appears in the main dashboard xml definition. The dashboard portlets/items are assigned an index according to the order in which they appear in the xml. The first item has an index of 1 e.g. setting this to 1 indicates that the first portlet/item in the main dashboard xml should be displayed. A tab can have any number of items.

24.3.3. Example Syntax

```
<DASHBOARD>

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

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

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

</DASHBOARD>
```

25. Jxls Reports

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

- Create an Excel file (either xls or xlsx) with any text, formulas and formatting that will be required in the final report. This is the Jxls template.
- Insert Jxls syntax to define where the raw data should go
- Create a new report in ART and select the Jxls template to use in the **Template** field

- If the report is of type **Jxls: ART Query**, the sql used to retrieve the data will be the one defined in the SQL source section. You can use parameters, rules or any other features of ART reports. In the template, use the identifier **results** to output report values. e.g.

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

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

- If the report is of type **Jxls: Template Query**, the sql used to retrieve the data will be the one defined in the template. The datasource used for the query will be the one selected in the datasource field of the report. Use the **jdbc.query** method to call the sql e.g. `jdbc.query('select * from cities')`. You can alternatively pass the datasource id of the datasource to use. In this case use the **jdbc.query2** method to call the sql e.g. `jdbc.query2(1, 'select * from cities')`. You can also use the datasource name e.g. `jdbc.query2('mydb', 'select * from cities')`.

- If the report uses parameters, avoid using the following names for the parameters as these are used for special purpose variables i.e. `results, jdbc, params, locale`.

25.1. Options

You can define a number of options in the **Options** field of the report.

Property	Data Type	Default	Description
areaConfigFile	String		The name of the file that has configuration details of areas within the template e.g. `jxls2config.xml`. This is an xml file that can be uploaded using the **Add files** option.

26. JPivot Reports

There are 3 report types that use the JPivot library to provide OLAP functionality.

26.1. Mondrian

This report type uses ART’s internal mondrian engine to run OLAP queries. To use it, do the following.

26.1.1. Create the schema xml file

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

26.1.2. Create a datasource

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

26.1.3. Create the report

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

26.1.4. Roles

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

26.2. Mondrian XMLA

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

26.2.1. Create the schema xml file

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

26.2.2. Create the datasources.xml file

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

26.2.3. Create a datasource

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

26.2.4. Create the report

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

26.2.5. Potential errors

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

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

26.2.6. Accessing the Mondrian XMLA instance from MS Excel

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

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

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

26.3. SQL Server XMLA

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

26.3.1. Configure SSAS for HTTP access

Some resources on how to do this are as follows

- http://geekswithblogs.net/darrengosbell/archive/2007/11/04/SSAS-Connecting-via-HTTP-on-Windows-Vista.aspx
- http://ssas-wiki.com/Articles#HTTP_Access

After you have finished the configuration, start IIS.

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

26.3.2. Provide access to the cube

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

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

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

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

26.3.3. Create a datasource

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

26.3.4. Create the report

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

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

Note:

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

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

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

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

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

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

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

27. Scheduling

In order to schedule a report, a user needs to have the **schedule_jobs** permission. Once logged in, a user’s scheduled reports are available from the **Jobs** menu.

27.1. Scheduling a new job

Take the following steps to schedule a new job

- From the Reports page, select the report to schedule and once the "select report parameters" page appears, click on the **Schedule** button. You can also schedule a job by going to the Reports Configuration page, finding the report to schedule and selecting the **More | Schedule** option.
- Specify the job details and Save
- The job will now run as per the schedule defined

Field	Description
ID	Auto-generated ID used to identify the job
Name	Name of the job
Job Type	The job type
Output Format	The output format for the job
Cached Datasource	For Cache ResultSet jobs, the datasource to use
Cached Table Name	For Cache ResultSet jobs, the table to use
Active	Whether the job can run or not
Enable Auditing	Whether job auditing is enabled. If disabled, only the last start and end time is recorded (in the `ART_JOBS` table). If enabled, an additional record is created in the `ART_JOBS_AUDIT` table indicating the start time, end time and status of each job run.
Number of Runs to Archive	For publish jobs, indicates how many runs should be retained for viewing from the Archives page e.g. setting it to 5 will retain output from the last 5 runs in the Archives page. This allows users to access past job output as per their requirements.
Allow Sharing	For publish jobs, whether the job output can be accessed by other users, apart from the job owner
Allow Splitting	Whether rule values for shared users should be applied so that each user gets different output

Field	Description
Fixed File Name	The base file name to be used for generated output. This base file name does not include the extension. The extension will be automatically provided depending on the report and output format. File names can only contain english alphabet letters, numbers, hyphen and space. Any other characters will be replaced with under-scores. Leave this field blank to use a default generated file name.
Destinations	Any destinations to which the generated output should be sent. Destinations are configured from the **Configure | Destinations** menu.
Sub-Directory	An additional sub-directory that should be used with destinations
Batch File	The name of a batch file or script that should be run after the job completes. The batch file must exist in the **WEB-INF\work\batch** directory. You only set the file name in this field e.g. `test.bat` or `test.sh`. ART passes the file name of the generated output as the first parameter to the batch file so you can use this e.g. to copy the file to another location, a batch file may contain the following command `-copy ..\export\jobs\%1 C:\temp`.
Pre Run Report	The report id of an Update Statement report that should be run before the job is run. Multiple reports can be specified separated by commas.
Post Run Report	The report id of an Update Statement report that should be run after the job has run. Multiple reports can be specified separated by commas.
From	The email address that will be used as the "From" email address of an email job
To	The email address to send job output to. Multiple addresses can be specified sepa-rated by `,`
Dynamic Recip-ients	A dynamic recipients report that specifies email addresses to send job output to, with the email addresses being obtained from a database query.
Cc	The "Cc" email address for a job. Multiple addresses can be specified separated by `,`
BCc	The "BCc" email address for a job. Multiple addresses can be specified separated by `,`
Subject	The subject of the email
Template	A thymeleaf template file that can be used as a template for the email body
SMTP Server	The smtp server to use, if different from the one configured in the **Settings** page
Message	The body contents of the email
Manual	Whether the job will run on an automatic schedule or not
Fixed Schedule	The schedule with which the job should run. If the fixed schedule is updated, the job's run schedule is also updated. If a fixed schedule is selected, the individual schedule fields are not considered.
Second	The second that the job should run. If not specified, defaults to 0.

Field	Description
Minute	The minute that the job should run. If not specified, defaults to a random minute (0-59).
Hour	The hour that the job should run. If not specified, defaults to a random hour between 3-6.
Month	The month that the job should run. If not specified, defaults to every month.
Day	The day of the month that the job should run i.e. 1-31. If not specified, defaults to every day.
Week Day	The day of the week that the job should run i.e. 1-7 or SUN-SAT. Note that you cannot specify both the **Month Day** and the **Week Day**. Specify one of the two, leaving the other one blank, or setting it to "?"
Year	The year that the job should run. If not specified, defaults to every year.
Time Zone	The time zone for the defined schedule
Start Date	The date when the job should start running. If blank, the job will start running on the current date, as per it's schedule.
End Date	The date when the job should stop running. If blank, the job will continue to run indefinitely.
Extra Schedules	Definitions of extra schedules on which the job should run, in addition to the main schedule defined in the individual schedule fields. Schedules can be defined as Quartz cron expressions, with each schedule on a new line. Alternatively, groovy code can be used, with the code returning a Trigger or a List of Triggers. If using groovy code, the field should start with "g[" and end with "]g".
Holidays	Defines dates or times on which the job should not run. These dates can be defined using quartz cron expressions as explained in the CronCalendar documentation, with each holiday definition on a new line. Alternatively, groovy code can be used, with the code returning a Calendar or a List of Calendars. If using groovy code, the field should start with "g[" and end with "]g".
Shared Holidays	Shared holidays configured from the **Configure | Holidays** menu that should be applied for this job.
Start Condition	A condition to be satisfied before a scheduled job is run
Error Notification Email	An email address that should be notified if an error occurs while running the job. Multiple email addresses can be specified separated by commas. The "From" email address will be that which is set in the **Settings** page under the **Error Notification** section. The SMTP server used is that configured in the Settings page.
Options	Any optional job options, specified in JSON format

Note:

- Not all report types can be scheduled
- If the output for a job creates a file e.g. publish to pdf, this file is stored in the **WEB-INF\work\export\jobs** directory

27.2. Job Options

Some options can be defined in the **Options** field. Options are specified in JSON format and include the following.

Property	Data Type	Default	Description
logInterval	Integer	0	For jobs that use dynamic recipients, this specifies an interval or number of records after which an item is included in the job log to indicate progress of the job. A value of 0 means this intermediate logging is not done.

27.3. Job Types

27.3.1. E-Mail Output (Attachment)

The output is emailed as an attachment to the specified recipients (in the "To" section). The attachment type can be selected in the **Output Format** field.

27.3.2. E-Mail Output (Inline)

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

27.3.3. Publish

The output is saved to a file. The file is reachable from the Jobs page. Once the file is generated, a notification email is sent to the specified recipients (in the "To" section). Leave the "To" area empty if you don’t want a notification email to be sent.

27.3.4. 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
```

27.3.5. Just Run It

Simply run the report. May be used to run stored procedures or perform data updates.

27.3.6. 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.

27.3.7. 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.

27.3.8. Conditional Publish

If the result set has records, the output will available from the Jobs page. A notification email is sent to the specified recipients if this is configured. If the result set has no records, no output is generated and no email is sent.

27.3.9. Cache ResultSet (Append)

Cache the result set in a database table (Append)

27.3.10. Cache ResultSet (Delete&Insert)

Cache the result set in a database table (Delete&Insert)

27.3.11. Print

Prints the report output to the default printer

27.3.12. Burst

Generates report files for each distinct value in a resultset. This requires that the query for the report be ordered by the first column, which will be the burst-id column. When the value of this column changes, a new file is generated. Burst jobs are only possible with **Tabular** reports. The files generated will have the burst-id in their file names so different files can be identified and copied or ftped to different locations for example.

27.4. Archives

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

27.5. Saved Schedules

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

27.6. 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, enable the **Allow Sharing** field and select the users or user groups with whom the output will be shared using the **Configure | Access Rights** menu. These users will be able to access the job's output from the Jobs menu.

27.6.1. Split Jobs

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

27.7. Random Start Times

You may not care when exactly a job runs, as long as it runs in a certain window of time. If you leave the Hour and Minute fields blank when creating the job, the job will be assigned a random hour between 3-6 and a random minute between 0-59 in which it will run. This may be useful for jobs that 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.

27.8. Quartz Properties

You can modify some aspects of the scheduler e.g. thread count by modifying the **WEB-INF\classes\quartz.properties** file. Documentation of configuration options can be found here. The following are example configurations.

Database Type	Quartz Configuration
Oracle	`org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.oracle.OracleDelegate org.quartz.dataSource.ArtDs.validationQuery = select 1 from dual`
DB2	`org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.StdJDBCDelegate org.quartz.dataSource.ArtDs.validationQuery = select 1 from sysibm.sysdummy1`
HSQLDB	`org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.HSQLDBDelegate org.quartz.dataSource.ArtDs.validationQuery = values 1`
PostgreSQL	`org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.PostgreSQLDelegate org.quartz.dataSource.ArtDs.validationQuery = select 1`
CUBRID	`org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.CUBRIDDelegate org.quartz.dataSource.ArtDs.validationQuery = select 1`
SQL Server	`org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.MSSQLDelegate org.quartz.dataSource.ArtDs.validationQuery = select 1`
Informix	`org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.StdJDBCDelegate org.quartz.dataSource.ArtDs.validationQuery = select 1 from systables where tabid = 1`
Firebird	`org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.StdJDBCDelegate org.quartz.dataSource.ArtDs.validationQuery = select 1 from RDB$DATABASE`

For other database types, you can supply an appropriate validation query as per your database. You don’t have to specify the driver delegate class. `StdJDBCDelegate` will be used.

28. Holidays

A holiday is a specification of dates or times when a job should not run. Use the **Configure | Holidays** menu to manage holidays.

Field	Description
ID	An auto-generate ID used to identify the holiday configuration
Name	A name for the holiday configuration
Description	A description for the holiday configuration
Definition	The holiday configuration. The dates or times can be specified using quartz cron expressions as explained in the CronCalendar documentation, with each date specification on a new line. Alternatively, groovy code can be used, with the code returning a Calendar or a List of Calendars. If using groovy code, the field should start with "g[" and end with "]g".

29. Destinations

You can configure destinations to which publish job output is sent. Use the **Configure | Destinations** menu to manage destination configurations.

29.1. Creating a new destination

When adding a destination, the following fields are available.

Field	Description
ID	An auto-generated ID used to identify the destination
Name	A name for the destination
Description	A description for the destination
Active	Whether the destination is active or not
Destination Type	The type of destination. The **Network Share** destination type will not work with SMB1 shares e.g. Windows XP or Windows Server 2003 shares.
Server	For **FTP** and **SFTP** destinations, the IP address or host name of the ftp or sftp server. For **Network Share** destinations, the IP address or hostname of the machine where the share is located.
Port	For **FTP** and **SFTP** destinations, the ftp or sftp port of the server. You can leave this as 0 to use the default port for the respective protocol.
User	The user to use to connect to the destination. For **Amazon S3** destinations, this is the IAM user’s access key id. For **Microsoft Azure** Blob Storage destinations, this is the storage account name. For **Backblaze B2** destinations, this is the Account ID.
Password	The destination user’s password. For **Amazon S3** destinations, this is the IAM user’s secret access key. For **Microsoft Azure** Blob Storage destinations, this is the primary access key. For **Backblaze B2** destinations, this is the Application Key.
JSON Key File	For **Google Cloud Storage** destinations, the service account JSON Key file
Domain	For **Network Share** destinations, an optional specification of the user’s domain
Path	The location of the destination. For **Network Share** destinations, this is the share name. For **FTP** and **SFTP** destinations, this is an optional directory path where files should be copied. For **Amazon S3**, **BackBlaze B2** and **Google Cloud Storage** destinations, this is the bucket name. For **Microsoft Azure** destinations, this is the container name. For **WebDAV** destinations, this is the URL of the WebDAV server e.g. `http://localhost`. For **Website** destinations, this is a URL that POSTs a form which includes a file field.
Sub-Directory	An optional sub-directory path within the destination where files should be copied
Create Directories	Whether directories in the path should be created before the file is copied. The user will need appropriate permissions for the creation of directories.
Options	Additional options for the destination

Note

- For **Amazon S3** destinations, the IAM user needs the **s3:GetBucketLocation** permission in order for the file upload to be successful. In addition, the user will need at least the **s3:PutObject** permission on the bucket where the file is to be copied.

29.2. Options

Some destinations can have additional options that can be specified. These are included in the **Options** field using JSON syntax.

29.2.1. FTP Options

For **FTP** destinations, the following options are available.

Attribute	Data Type	Default	Description
connectTimeoutSeconds	Integer	60	The timeout (in seconds) to use when connecting to the server.
controlKeepAliveTimeoutSeconds	Long		The time period (in seconds) after which to send a keep alive (NOOP) message. Zero disables.

29.2.2. SFTP Options

For **SFTP** destinations, the following options are available.

Attribute	Data Type	Default	Description
sessionConnectTimeoutSeconds	Integer	0	The timeout (in seconds) to use when opening a session. Zero indicates "no timeout".

29.2.3. Network Share Options

For **Network Share** destinations, the following options are available.

Attribute	Data Type	Default	Description
timeoutSeconds	Integer	60	The timeout (in seconds) for read/write operations.
multiProtocolNegotiate	Boolean	false	For suspected SMB1 shares, this can be enabled in order to get an error message indicating that SMB1 is not supported.

29.2.4. Website Options

For **Website** destinations, the following options are available.

Attribute	Data Type	Default	Description
loginUrl	String		A url that takes a POST request that will be used to log in before uploading the file
startUrl	String		A url that takes a GET request that should be visited first before uploading the file
fileField	String	file	The name of the html field for the file that is being submitted
usernameField	String	username	When using a **loginUrl**, this is the name of the html field that contains the username to use to authenticate
passwordField	String	password	When using a **loginUrl**, this is the name of the html field that contains the password to use to authenticate
csrfTokenCookie	String		The name of a cookie that contains a csrf token that needs to be included when submitting the file. This is used either with a **startUrl** or **loginUrl** in order to obtain the csrf token.
csrfTokenInputField	String		The name of a html input element that contains a csrf token that needs to be included when submitting the file. This is used either with a **startUrl** or **loginUrl** in order to obtain the csrf token.
csrfTokenOutputField	String		The name of a html field that contains a csrf token when submitting the file
staticFields	Array of objects		An array of key-value pairs representing fields that contain static data that should be included when submitting the file. The keys are the html field names and values are the field values e.g. `"staticFields": [{"field1": "value1"}, {"field2": "value2"}]`.

29.2.5. Amazon S3 - AWS SDK Options

For **Amazon S3 - AWS SDK** destinations, the following options are available.

Attribute	Data Type	Default	Description
region	String	us-east-1	The region that will process the requests. This is not necessarily the region where the bucket is located. You can set a region near where the ART server is in order to get faster operation. See the s3 regions documentation for a list of possible region values.
cannedAcl	String		A canned acl that should be applied to the uploaded file. The possible strings are as per the CannedAccessControlList class i.e. either **AuthenticatedRead**, **AwsExecRead**, **BucketOwnerFullControl**, **BucketOwnerRead**, **LogDeliveryWrite**, **Private**, **PublicRead** or **PublicReadWrite**. See the canned acl documentation for more details about canned acl options.

30. SMTP Servers

You can configure smtp servers to be used by individual jobs. You may want a job to use a different smtp server from the one configured in the **Settings** page.

Use the **Configure | SMTP Servers** menu to manage smtp server configurations. When adding an smtp server, the following fields are available.

Field	Comment
ID	An auto-generated ID used to identify the smtp server configuration
Name	A name for the smtp server configuration
Description	A description for the smtp server configuration
Active	Whether this configuration is active or not. If it is not active, jobs that use this smtp server will not send emails.
Server	The IP address or host name of the smtp server
Port	The smtp port to use
Use StartTLS	Whether to use STARTTLS when connecting to the server
Use SMTP Authentication	Whether to use a username and password when connecting to the smtp server
Use Google OAuth2	Whether to use OAuth when using a Gmail smtp server
Username	The username to use to connect to the smtp server, typically an email address on the server
Password	The password to use to connect to the smtp server
OAuth Client ID	When using OAuth, the Client ID
OAuth Client Secret	When using OAuth, the Client Secret
OAuth Refresh Token	When using OAuth, the Refresh Token
From	The "From" email address for jobs that will use this smtp configuration. This can be left blank in which case individual "From" email addresses specified within jobs will be used.

30.1. Using Gmail

When using Gmail as an smtp server, specify **smtp.gmail.com** as the server, **587** as the port, enable **Use StartTLS** and enable **Use SMTP Authentication**. You have two options for specifying the credentials to use.

30.1.1. Using username and password

You can specify the email address in the username field and the email account password in the password field. In order for the connection to be successful, you'll need to change the security settings for the account to enable the **Allow less secure apps** option. Some resources on how to do this include the following.

- https://support.google.com/accounts/answer/6010255
- https://dzone.com/articles/sending-mail-using-javamail-api-for-gmail-server
- https://www.dev2qa.com/how-do-i-enable-less-secure-apps-on-gmail/

If you don't enable Allow less secure apps and you use the username and password to connect, you will get an error like **javax.mail.AuthenticationFailedException: 535-5.7.8 Username and Password not accepted.**

30.1.2. Using OAuth

Instead of using the username and password for the gmail account, you can configure the account to allow use of OAuth when sending emails. You can follow the instructions in this article to guide you on creating the OAuth Client ID, Client Secret and Refresh Token that will be required in the configuration. Important to note that the **scope** to be used when creating the tokens is `https://mail.google.com/`, as per the Google documentation. Note also that you will still need to put the gmail email address in the **Username** field.

Note:

- Some antivirus programs may block java applications from sending emails. If you get **Connection refused** errors, you can try and temporarily disable the antivirus program on the server and see if the connection will be successful. If so, find out how to add exceptions in the antivirus program and add an exception for the **java.exe** file in use. For Avast Antivirus, you may need to turn off the Scan outbound emails(SMTP) option in the Mail Shield.

31. Cached Results

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

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

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

31.1. Create a new Cached Result

In order to cache the results of an existing report:

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

31.2. Accessing a Cached Result

At the given time ART will:

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

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

Note:

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

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

32. Dynamic Recipients

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

Using the dynamic recipients feature will involve the following process

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

There are several ways to use dynamic recipients.

32.1. Dynamic Recipients only

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

```
SELECT email
FROM employee
```

32.2. Dynamic Recipients + Personalization

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

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

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

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

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

32.3. Dynamic Recipients + Filtering

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

If the values in the **recipient_id** column are numbers, include an additional hard coded column named **recipient_id_type**, with the value of this column being the string "**number**". You can include columns named **open_password** and **modify_password** to contain dynamic passwords that should be used for the dynamic output if using xlsx or pdf report formats.

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, "number" recipient_id_type
FROM employee
```

This will cause the `#recipient#` placeholder in the data query to be replaced with the condition `transaction_user = <employee_id>`, where `<employee_id>` will be different for each recipient, as per the dynamic recipients query. We can then schedule a job for the main report and set the **Dynamic Recipients** field to the dynamic recipients report, 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, "number" recipient_id_type, first_name
FROM employee
```

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

Note:

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

33. Pipelines

A pipeline is a definition of jobs that should be run in a particular order. Use the **Configure | Pipelines** menu to configure pipelines.

Field	Description
ID	An auto-generate ID used to identify the pipeline
Name	A name for the pipeline
Description	A description for the pipeline
Serial	A comma separated list of job ids that should be run in the order given. You can use **all** to specify that all jobs should be run. You can also specify jobs that use a given schedule by using the syntax **schedule:**`<schedule name>` e.g. `schedule: daily`. You can also use `+` after a job id to indicate running of all jobs from that id to the last e.g. `13+`. You can also use `-` to specify job ranges e.g. `1-5`. You can specify jobs whose report belongs to a certain report group by using the syntax **report-Group:**`<report group name>` e.g. `reportGroup: test`. You can specify multiple report groups by separating them with `;` e.g. `reportGroup: test;admin`.
Continue On Error	For serial jobs, whether the next jobs should be run if the current one encounters an error
Schedule	A schedule on which the pipeline will run
Start Condition	A condition to be satisfied before a scheduled pipeline is run

34. Start Conditions

A start condition is a definition of a condition to be checked before a scheduled job or pipeline runs. Use the **Configure | Start Conditions** menu to configure start conditions.

Field	Description
ID	An auto-generate ID used to identify the start condition
Name	A name for the start condition
Description	A description for the start condition
Retry Delay (Mins)	The amount of time to wait in minutes if the condition is not fulfilled, before making the next try
Retry Attempts	The number of times to retry the condition
Condition	The condition to check for before running the job or pipeline. This could either be an integer representing a report id, or groovy code which would return true or false. For a report id, the first column of the result needs to be an integer and if it's greater than 0, then the condition will be considered as fulfilled and the job will run. For groovy code, if the return result is **true**, then the condition will be considered as fulfilled and the job will run. If the condition is not fulfilled, it will be tried again after the delay interval. After the retry attempts are exhausted and the condition is still false, the job or pipeline will not be run.

35. Integrated Windows Authentication

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

35.1. 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)

35.2. 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
```

35.3. 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>
```

35.4. On the client machine

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

35.4.1. Omitting the credentials box

- **Firefox**
  By default, firefox will still display a credentials box requiring a user to enter their domain user-name and password. To avoid this, do the following

  - In the address bar, type **about:config**
  - In the filter box, type **network.negotiate**
  - Double click on the **network.negotiate-auth.trusted-uris** option and enter the url of the application server (excluding the http part and including port number if not port 80) e.g. app-server:8080

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

  - **network.negotiate-auth.delegation-uris**
  - **network.automatic-ntlm-auth.trusted-uris**

- **IE**
  For IE, the credentials box may be displayed if you access the web application using an ip address. To avoid this, do the following

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

35.5. Using a keytab file

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

- Stop tomcat
- Copy the file ART_HOME\WEB-INF\krb5.conf to the windows directory e.g. C:\windows
- Rename the file to krb5.ini

- Open a command prompt window, cd to the ART_HOME\WEB-INF directory and type a command with syntax like the following

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

E.g.

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

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

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

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

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

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

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

- Restart tomcat

You should have integrated authentication as before

35.6. 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
```

36. CAS Authentication

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

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

37. API

A REST API is provided to facilitate interacting with some aspects of ART in a programmatic fashion, as opposed to using the browser.

37.1. Base URL

The base URL for all api calls is **/art/api/**

37.2. Responses

Calls to the api will result in responses that contain an **ApiResponse** object in the response body. This object has the following fields.

Field	Data Type	Description
httpStatus	Integer	The HTTP status code of the response
artStatus	String	An indication of the status of the action. **OK** means that the operation was successful, **ERROR** indicates a server or HTTP error, and other values indicate that the operation was not done due to some kind of problem.
message	String	A descriptive message e.g. indicating the problem with a request
data	Object	Data relevant for the response

37.2.1. HTTP Statuses

The following HTTP statuses are used.

Status	Description
200	The operation was successful
201	An add operation was successful. The **Location** header will contain the url of the newly created resource.
400	There was an invalid value in the request
401	No authorization details were provided or authorization could not be granted given the credentials presented
409	A delete could not be perfomed because related records exist or a new record could not be added because a similar record already exists
500	An error occurred while performing the request

37.2.2. ART Statuses

The **artStatus** field of the **ApiResponse** can have a number of values as below.

Status	Description
OK	The operation was successful
ERROR	There was a server or HTTP error
AUTHORIZATION_REQUIRED	There was no **Authorization** header in the request. Calls to the **/api/login** endpoint don't require an Authorization header. All other endpoints require this header.
UNAUTHORIZED	The operation could not be perfomed because authorization is not available. This may mean that the user whose credentials have been supplied does not exist, is disabled or doesn't have the **use_api** permission. It may also mean that token based authentication is not configured i.e. the **jwtSecret** field in the **WEB-INF\art-custom-settings.json** file has not been set.
RECORD_NOT_FOUND	The referenced record doesn't exist e.g. if trying to retrieve a user's details using the **/api/users/{id}** endpoint.
LINKED_RECORDS_EXIST	Indicates that a delete operation could not be perfomed because some related records exist e.g. you can't delete a user if that user is the owner of some jobs.
RECORD_EXISTS	Indicates that an add operation could not be perfomed because a similar record already exists e.g. if trying to add a user but another user with the same username already exists.
INVALID_VALUE	Indicates that some value provided in the request body is not appropriate e.g. trying to add a user with an empty username.

37.3. Authentication

All calls to the api must contain authentication information. This is contained in the **Authorization** HTTP header. API authentication always uses the Internal authentication method even though ART may be configured to use other authentication methods.

37.3.1. Basic Authentication

One can use the HTTP Basic authentication method to provide authentication credentials where the Authorization header cotains the username and password encoded in base 64 format i.e. `Authorization: Basic <base 64 credentials>`

37.3.2. Bearer Authentication

One can also obtain a token from the application and supply that with subsequent api calls. The **jwtSecret** in the **WEB-INF\art-custom-settings.json** file must have been set to a non-blank string. To obtain a token, perform a **POST** to the **/api/login** endpoint providing **username** and **password** form parameters. If authenticated, you will receive a response that contains an access token. This token should then be provided in the Authorization header using the **Bearer** scheme i.e. `Authorization: Bearer <token>`. By default, tokens expire after 60 minutes. This can be changed from the **Settings** page using the **JWT Token Expiry (Mins)** field.

37.4. Login

37.4.1. Request

If using bearer authentication, you obtain an access token by performing the following request, passing **username** and **password** form parameters

```
POST /api/login
```

37.4.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": {
        "accessToken": "<token>"
    }
}
```

In case the **jwtSecret** is not set, or the user provided doesn't exist or is disabled or doesn't have the **use_api** permission, a 401 response like the following would be returned

```
{
    "httpStatus": 401,
    "artStatus": "UNAUTHORIZED",
    "message": null,
    "data": null
}
```

37.5. Users

37.5.1. Getting all users

37.5.1.1. Request

```
GET /api/users
```

37.5.1.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": [
        {
            "userId": 24,
            "username": "admin",
            "accessLevel": "SuperAdmin",
            "email": "admin@localhost.local",
            "fullName": "",
            "password": null,
            "passwordAlgorithm": null,
            "startReport": "",
            "active": true,
            "canChangePassword": true,
            "creationDate": 1511116137000,
            "updateDate": null,
            "userGroups": [],
            "effectiveStartReport": null,
            "effectiveDefaultReportGroup": null,
            "useBlankPassword": false,
            "createdBy": "admin",
            "updatedBy": null,
            "generateAndSend": false,
            "clearTextPassword": false,
            "defaultReportGroup": null,
            "adminUser": true,
            "publicUser": false
        }
    ]
}
```

37.5.2. Getting a particular user

37.5.2.1. Request

Get by user id

```
GET /api/users/{id}
```

Get by username

```
GET /api/users/username/{username}
```

37.5.2.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": {
            "userId": 1,
            "username": "admin",
            "accessLevel": "SuperAdmin",
            "email": "admin@localhost.local",
            "fullName": "",
            "password": null,
            "passwordAlgorithm": null,
            "startReport": "",
            "active": true,
            "canChangePassword": true,
            "creationDate": null,
            "updateDate": 1514299510000,
            "userGroups": [],
            "effectiveStartReport": "",
            "effectiveDefaultReportGroup": null,
            "useBlankPassword": false,
            "createdBy": null,
            "updatedBy": "admin",
            "generateAndSend": false,
            "clearTextPassword": false,
            "defaultReportGroup": null,
            "adminUser": true,
            "publicUser": false
        }
    }
```

In case the user doesn't exist, a 404 response like the following would be returned

```
    {
        "httpStatus": 404,
        "artStatus": "RECORD_NOT_FOUND",
        "message": null,
        "data": null
    }
```

37.5.3. Deleting a user

37.5.3.1. Request

```
    DELETE /api/users/{id}
```

37.5.3.2. Response

If successful, a 200 response like the following would be returned

```
    {
        "httpStatus": 200,
        "artStatus": "OK",
        "message": null,
        "data": null
    }
```

In case related records exist, a 409 response like the following would be returned. The data attribute would contain a list of strings which represent the jobs that the user owns.

```
{
    "httpStatus": 409,
    "artStatus": "LINKED_RECORDS_EXIST",
    "message": "User not deleted because linked jobs exist",
    "data": [
        "A report (5)",
        "Pie (8)",
    ]
}
```

37.5.4. Adding a user

37.5.4.1. Request

```
POST /api/users
    {
        "username": "test",
        <other fields>
    }
```

The request body should contain a user object. You can use JSON format and set the **Content-Type** header to **application/json**. For the applicable field names of the **User** object, you can look at the **WEB-INF\classes\art\user\User.java** file. If using a clear text **password**, include the **clearTextPassword** field and set it to **true**.

37.5.4.2. Response

If successful, a 201 response like the following would be returned. The **Location** header in the response will contain the url for the newly created user. The created user is included in the response.

```
{
    "httpStatus": 201,
    "artStatus": "OK",
    "message": null,
    "data": {
        "userId": 41,
        <other fields>
    }
}
```

If the username is not provided or is blank, a 400 response like the following would be returned

```
{
    "httpStatus": 400,
    "artStatus": "INVALID_VALUE",
    "message": "username field not provided or blank",
    "data": null
}
```

If a user with the supplied username already exists, a 409 response like the following would be returned

```
{
    "httpStatus": 409,
    "artStatus": "RECORD_EXISTS",
    "message": "A user with the given username already exists",
    "data": null
}
```

37.5.5. *Updating a user*

37.5.5.1. Request

```
PUT /api/users/{id}
    {
        "username": "test"
        <other fields>
    }
```

The request body should contain a user object. You can use JSON format and set the **Content-Type** header to **application/json**. For the applicable field names of the **User** object, you can look at the **WEB-INF\classes\art\user\User.java** file. If using a clear text **password**, include the **clearTextPassword** field and set it to **true**.

37.5.5.2. Response

If successful, a 200 response like the following would be returned. The updated user is included in the response.

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": {
        "userId": 12,
        <other fields>
    }
}
```

37.5.6. *Disabling a user*

37.5.6.1. Request

```
POST /api/users/{id}/disable
```

37.5.6.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": null
}
```

37.5.7. *Enabling a user*

37.5.7.1. Request

```
POST /api/users/{id}/enable
```

37.5.7.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": null
}
```

37.6. Reports

37.6.1. Getting a particular report

37.6.1.1. Request

Get by id

```
GET /api/reports/{id}
```

Get by name

```
GET /api/reports/name/{name}
```

37.6.1.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": {
        "reportId": 1,
        "name": "test",
        "shortDescription": "short desc",
        "description": "long desc",
        "contactPerson": "contact",
        "usesRules": false,
        "parametersInOutput": false,
        "xAxisLabel": "",
        "yAxisLabel": "",
        "chartOptionsSetting": null,
        "template": "",
        "displayResultset": 0,
        "xmlaDatasource": "",
        "xmlaCatalog": "",
        "creationDate": null,
        "updateDate": 1560255530000,
        "reportSource": "select * from mytable",
        "createdBy": null,
        "createdById": 0,
        "updatedBy": "admin",
        "reportType": "Tabular",
        "groupColumn": 0,
        "active": true,
            "hidden": false,
            "defaultReportFormat": "default",
            "secondaryCharts": "",
            "hiddenColumns": "",
            "totalColumns": "",
            "dateFormat": "",
            "numberFormat": "",
            "columnFormats": "",
            "locale": "",
            "nullNumberDisplay": "",
            "nullStringDisplay": "",
            "fetchSize": 0,
            "options": "",
            "pageOrientation": "Portrait",
            "omitTitleRow": false,
            "lovUseDynamicDatasource": false,
            "openPassword": null,
            "modifyPassword": null,
            "sourceReportId": 0,
            "reportGroups": [
                {
                    "parentId": 0,
                    "reportGroupId": 2,
                    "name": "Test",
                    "description": "test group",
                    "creationDate": null,
                    "updateDate": null,
                    "createdBy": null,
                    "updatedBy": null
                }
            ],
            "clearTextPasswords": false,
            "useGroovy": false,
            "pivotTableJsSavedOptions": "",
            "gridstackSavedOptions": "",
            "name2": null,
            "comment": "",
            "viewReportId": 0,
            "selfServiceOptions": "",
            "datasource": null,
            "encryptor": null,
            "reportParams": null,
            "userRuleValues": null,
            "userGroupRuleValues": null,
            "reportRules": null,
            "userReportRights": null,
            "userGroupReportRights": null,
            "drilldowns": null,
            "description2": null,
            "dtActiveStatus": null,
            "dtAction": null,
            "reportGroupNames": "Test",
            "reportGroupNamesHtml": "Test",
            "dtId": 1,
            "dtName": "test"
        }
    }
```

In case the report doesn’t exist, a 404 response like the following would be returned

```
{
    "httpStatus": 404,
    "artStatus": "RECORD_NOT_FOUND",
    "message": null,
    "data": null
}
```

37.6.2. Deleting a report

37.6.2.1. Request

```
DELETE /api/reports/{id}
```

37.6.2.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": null
}
```

In case related records exist, a 409 response like the following would be returned. The data attribute would contain a list of strings which represent the jobs that use the report.

```
{
    "httpStatus": 409,
    "artStatus": "LINKED_RECORDS_EXIST",
    "message": "Report not deleted because linked jobs exist",
    "data": [
        "A report (5)",
        "Pie (8)",
    ]
}
```

37.6.3. Adding a report

37.6.3.1. Request

```
POST /api/reports
    {
        "name": "test",
        <other fields>
    }
```

The request body should contain a report object. You can use JSON format and set the **Content-Type** header to **application/json**. For the applicable field names of the **Report** object, you can look at the **WEB-INF\classes\art\report\Report.java** file.

37.6.3.2. Response

If successful, a 201 response like the following would be returned. The **Location** header in the response will contain the url for the newly created report. The created report is included in the response.

```
{
    "httpStatus": 201,
    "artStatus": "OK",
    "message": null,
    "data": {
        "reportId": 41,
        <other fields>
    }
}
```

If the name is not provided or is blank, a 400 response like the following would be returned

```
{
    "httpStatus": 400,
    "artStatus": "INVALID_VALUE",
    "message": "name field not provided or blank",
    "data": null
}
```

If a report with the supplied name already exists, a 409 response like the following would be returned

```
{
    "httpStatus": 409,
    "artStatus": "RECORD_EXISTS",
    "message": "A report with the given name already exists",
    "data": null
}
```

37.6.4. Updating a report

37.6.4.1. Request

```
PUT /api/reports/{id}
    {
        "name": "test",
        <other fields>
    }
```

The request body should contain a report object. You can use JSON format and set the **Content-Type** header to **application/json**. For the applicable field names of the **Report** object, you can look at the **WEB-INF\classes\art\report\Report.java** file.

37.6.4.2. Response

If successful, a 200 response like the following would be returned. The updated report is included in the response.

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": {
        "reportId": 12,
        <other fields>
    }
}
```

37.6.5. Disabling a report

37.6.5.1. Request

```
POST /api/reports/{id}/disable
```

37.6.5.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": null
}
```

37.6.6. Enabling a report

37.6.6.1. Request

```
POST /api/reports/{id}/enable
```

37.6.6.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": null
}
```

37.6.7. Running a report

37.6.7.1. Request

```
POST /api/reports/run
```

As part of the request, form parameters need to be included to provide parameters to be used to run the report, similar to running a report via url in the browser. The **reportId** parameter is mandatory in order to specify which report to run e.g. `.../api/reports/run?reportId=1`. Html report formats are not supported when running a report via api.

37.6.7.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": {
        "fileName": "report1-2019_06_20-10_50_52_700-WP985-1-0.pdf",
        "url": "http://localhost:8080/art/export/reports/report1-2019_06_20-10_50_52_700-WP985-1-0.pdf",
        "rowsRetrieved": 12,
        "rowsUpdated": null
    }
}
```

If the report is not found, a 404 response like the following would be returned

```
{
    "httpStatus": 404,
    "artStatus": "RECORD_NOT_FOUND",
    "message": null,
    "data": null
}
```

If a html report format is specified, a 400 response like the following would be returned

```
{
    "httpStatus": 400,
    "artStatus": "INVALID_VALUE",
    "message": "report format not allowed: html",
    "data": null
}
```

37.7. User Groups

37.7.1. Getting all user groups

37.7.1.1. Request

```
GET /api/user-groups
```

37.7.1.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": [
        {
            "userGroupId": 4,
            <other fields>
        }
    ]
}
```

37.7.2. Getting a particular user group

37.7.2.1. Request

Get by id

```
GET /api/user-groups/{id}
```

Get by name

```
GET /api/user-groups/name/{name}
```

37.7.2.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": {
        "userGroupId": 1,
        <other fields>
    }
}
```

In case the user group doesn't exist, a 404 response like the following would be returned

```
{
    "httpStatus": 404,
    "artStatus": "RECORD_NOT_FOUND",
    "message": null,
    "data": null
}
```

37.7.3. Deleting a user group

37.7.3.1. Request

```
DELETE /api/user-groups/{id}
```

37.7.3.2. Response

If successful, a 200 response like the following would be returned

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": null
}
```

In case related records exist, a 409 response like the following would be returned. The data attribute would contain a list of strings which represent the users that the user group contains.

```
{
    "httpStatus": 409,
    "artStatus": "LINKED_RECORDS_EXIST",
    "message": "User Group not deleted because linked users exist",
    "data": [
        "user1",
        "test",
    ]
}
```

37.7.4. *Adding a user group*

37.7.4.1. Request

```
POST /api/user-groups
    {
        "name": "test",
        <other fields>
    }
```

The request body should contain a user group object. You can use JSON format and set the **Content-Type** header to **application/json**. For the applicable field names of the **UserGroup** object, you can look at the **WEB-INF\classes\art\usergroup\UserGroup.java** file.

37.7.4.2. Response

If successful, a 201 response like the following would be returned. The **Location** header in the response will contain the url for the newly created user group. The created user group is included in the response.

```
{
    "httpStatus": 201,
    "artStatus": "OK",
    "message": null,
    "data": {
        "userGroupId": 5,
        <other fields>
    }
}
```

If the name is not provided or is blank, a 400 response like the following would be returned

```
{
    "httpStatus": 400,
    "artStatus": "INVALID_VALUE",
    "message": "name field not provided or blank",
    "data": null
}
```

If a user group with the supplied name already exists, a 409 response like the following would be returned

```
{
    "httpStatus": 409,
    "artStatus": "RECORD_EXISTS",
    "message": "A user group with the given name already exists",
    "data": null
}
```

37.7.5. Updating a user group

37.7.5.1. Request

```
PUT /api/user-groups/{id}
    {
        "name": "test",
        <other fields>
    }
```

The request body should contain a user group object. You can use JSON format and set the **Content-Type** header to **application/json**. For the applicable field names of the **UserGroup** object, you can look at the **WEB-INF\classes\art\usergroup\UserGroup.java** file.

37.7.5.2. Response

If successful, a 200 response like the following would be returned. The updated user group is included in the response.

```
{
    "httpStatus": 200,
    "artStatus": "OK",
    "message": null,
    "data": {
        "userGroupId": 10,
        <other fields>
    }
}
```

38. 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 **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**. Scan period 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 **View | Logs** menu. New logs are added to the bottom and you'll need to refresh the page to view them. Also note that this page doesn't display all the logs ever generated by the application, only the most recent ones.

38.1. SQL Logging

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

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

- Modify the **WEB-INF\classes\logback.xml** file and add loggers for the items you are interested in e.g.

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

- The available loggers and what events they log can be found on the log4jdbc-log4j2 project home page, http://log4jdbc.brunorozendo.com/. ART uses the SLF4J configuration.

- That's all. The logging information will now be included in the application logs when queries are run against that datasource.

39. Tomcat Configuration

39.1. 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.

39.1.1. Windows

If Tomcat was installed as a service

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

Note:

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

If Tomcat is run from a batch file

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

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

39.1.2. Linux

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

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

39.2. 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.

39.2.1. 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`).

39.2.2. 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`).

39.2.3. 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.

40. Customizing ART

You can customize ART to fit your needs.

40.1. 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.

40.1.1. 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`

40.1.2. 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 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

40.1.3. Customizing jsp files

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

40.1.4. Customizing other files

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

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

40.1.5. Updating jasperreports

If you are using a version of Jaspersoft Studio that is different from the jasperreports version contained in ART, you may want to update the jasperreports version contained in ART to match your Jaspersoft Studio 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 Ant is set up as described above
- Stop Tomcat
- Backup the contents of the **ART_HOME\WEB-INF\work\templates** directory, where the jasperreport template files reside
- Delete the jasperreports .jar file contained in the **ART_HOME\WEB-INF\lib** directory
- Copy the just downloaded jasperreports .jar file to the **ART_HOME\WEB-INF\lib** directory
- Open a command prompt window and navigate to the ART_HOME directory
- Type the command `ant clean-jasperreports`
- Type the command `ant compile-jasperreports`
- Restart Tomcat

Your reports should now work properly.

40.2. 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.

40.2.1. Using Ant without an IDE

- Ensure 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 clean compile war`
- A new file named art.war will be created in the `PACKAGE_PATH\art` directory. Deploy this file to your desired application server.

40.2.2. Using Ant with NetBeans

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

- Unzip the `PACKAGE_PATH\art.war` file e.g. unzip to a new directory `PACKAGE_PATH\art`. You can unzip the war file the same way as any zip file e.g. using 7zip.
- Create a directory to hold NetBeans project files e.g. `C:\MyNetBeansProjects\art`
- Open NetBeans
- Select the **File | New Project** menu
- Select the **Java Web** category and **Web Application with Existing Sources** project and click on the **Next** button
- For the **Location** field, select the `PACKAGE_PATH\art` directory. For the Project Folder field, select the directory you created for this e.g. `C:\MyNetBeansProjects\art`. Click on the **Next** button.
- Select the Server to use e.g. Apache Tomcat and Java EE version e.g. Java EE 7 Web. Click on the **Next** button.
- The Existing Sources and Libaries 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

40.3. Using Maven

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

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

40.3.1. 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.

40.3.2. 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

40.3.3. Using Maven with Eclipse

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

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

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

- Right click on the art-parent project and select the **Run As | Run Configurations** menu. Select the **Maven Build** option, and then click on the **New launch configuration** icon at the top of the list. Give a **Name** to the configuration e.g. package and set the **Base directory** to the `PACKAGE_PATH\src\art-parent` directory. In the **Goals** field, type `clean package`. In the JRE tab, ensure the JRE selected is the one defined earlier. Click on the Apply button. To generate a war file immediately, click on the Run button.
- To test changes, set up a server. See https://github.com/OneBusAway/onebusaway/wiki/Setting-Up-a-Tomcat-Server-in-Eclipse . Convert the art webapp project to a Dymanic 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

40.4. Translating ART

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

40.4.1. Online

ART has a translation project on the Crowdin platform. You can create an account on Crowdin and join the ART translation project at the following link. https://crowdin.com/project/artreporting. You can then translate online. If your language is not available, create a post on the ART sourceforge forum requesting for the new language to be added to the Crowdin project.

40.4.2. Offline

- Download the latest ART package, unzip it and unzip the **art.war** file using any zip/unzip software. Get the **ArtMessages.properties** file from the **ART_HOME\WEB-INF\i18n** directory and and make a copy of it, naming your new file in the format **ArtMessages_xx.properties**, where xx is the ISO 639-1 language code for your language. A List of these language codes can be found here. An example would be ArtMessages_pt.properties for Portuguese. If your language is written differently in different countries you can add the 2 letter country specifier to the file name so that you have a file name like ArtMessages_pt_BR.properties for Brazilian Portuguese. You can find a list of country codes here.
- Change all the text in your new properties file after the = signs to your language

- If your language uses a non ISO-8859-1 (Latin-1) character set (e.g. chinese, arabic, russian etc) name the file ArtMessages_xx-UTF8.properties and then decode it to the ISO-8859-1 character set with UTF-8 escapes using the **native2ascii** tool that is available with your Java installation. An example command is as below.

```
native2ascii -encoding utf-8 ArtMessages_xx-UTF8.properties ArtMessages_xx.properties
```

- Once you have your translated properties file in the i18n folder, you can restart the application server to make the new translation available to the application, or wait for 1 hour after which the messages are refreshed.

- Now you can log in to the application and include the **lang=xx** parameter in browser url to activate the new language e.g. `http://localhost:8080/art/reports?lang=de`
- If you would like the new language to be listed for selection within the application e.g. on the login page, add the language code and the language name to the **ART_HOME\WEB-INF\i18n\languages.properties** file, save the file and restart the application server.
- If in addition you would like this new translation to be included in future ART versions, email the properties file to **tanyona** (at) **users.sf.net** or create a new post on the Discussion forum and attach the file there.

41. Support

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

41.1. Commercial Support

For commercial support, email timothy.anyona (a t) halogen.co.ke