# Overview

This document provides a generic model for basic integration of a new vendor’s reporting system into ActiveNet. This model is based on the current integration with JReports, and is also the model which was developed for integrating Tableau with ActiveNet.

Based on information obtained from the vendors so far, it should also work with Jasper and Pentaho. Therefore, when we are doing our technical integration review with these vendors, we’ll use this model to review whether their capabilities are adequate. If there is a gap, we will need to explore alternative implementations with them, but this model will be the starting point.

# Assumptions

For the purpose of this document, some basic assumptions are made:

* Reports are launched from a new application, the “report portal”, which obtains its configuration information for users and reports from the servlet. However, if R1 does not have a separate portal, all of the other design points below work unchanged.
* The reporting system will be primarily responsible for displaying any filter UI.
* Reports get their data from a multi-tenant ADP store.

# Reference integration model



## 1. Developers create semantic models and template reports

In order to support ad hoc report generation by users, these products all provide tools for generating semantic models to abstract the actual data model; JReports calls these “business cubes”, Jasper calls them “domains”, etc. These tools and the resulting models would be created only by Active staff.

For each of these models, one or more sample reports will also be built by Active staff, usable out of the box, but also as a starting point (template) for customized versions built by the org.

After standalone testing in the development tool, the models and reports would be exported as files and checked into a repository, from which integrated testing environments and ultimately build images for deployment to production environments would be built.

## 2. Reports are published into production (QA, etc.) environment

When a build image containing the models and reports is deployed to a production environment, the deployment process must do the following operations against the report server in production, using the server’s API or other automation support:

* For each (new) org:
  + Create a folder for that org’s custom reports
  + Create a user for that org; depending on the details of the security model established, this could be a few generic users per org (e.g., users with and without the ability to save customized reports).
  + Create the permissions to allow these users to have access to appropriate folders.
* Publish the template reports to a location shared by all orgs.
* Depending on the selected report server, models may also be published to the server.

## 3. Reporting service stores reports in folders

It’s assumed based on capabilities shared by all these projects that it is possible to store models in something like folders, to which permissions can be attached. These don’t need to support a hierarchy (e.g., Tableau’s “projects” did not). The minimum folders needed are:

* A “public” folder containing the models and template reports available to all orgs.
* One folder per org containing its customized reports.

## 4. Reporting service has one (or a few) users per org, granting folder access

It’s also assumed that there is a capability to have create users in the report server, with permissions on the folders.

* In the simplest model, there would be one user per org. This user would have:
  + Read permissions on the public folder.
  + Read and write permissions on the org’s folder.
* In addition, it is assumed to be necessary to control which ANet users have the ability to make custom reports. Depending on the report server, it may be possible to control this at the time the report is launched, via URL parameters, as Tableau might have supported. If not, different reporting roles would be implemented by additional role-specific users for an org. At the minimum, there would have non-customizer roles, whose permissions would be:
  + Read permissions on the public folder.
  + Read permissions on the org’s folder.

There would be a naming convention to generate the names of these users from the org (and possibly reporting role).

## 5. Servlet uses report service API to enumerate reports available for org

Because org users can create additional reports for the org, which then need to be displayed in the report launch UI and have ANet user permissions applied to them, it must use a report server API to enumerate the reports available for that org, both template and customized.

The report data will be stored in a table in the org’s database, and provided to the portal via web services.

## 6. Servlet provides agency UI to control report permissions for ANet users

In the minimal model earlier proposed for an R1 implementation with Tableau, this will be extremely limited:

* There will be a “license” setting to turn on the new reporting system for a given org.
* There will be two new user profile option giving a specific user access to the new reporting system:
  + A read-only permission (“reader”)
  + A read-write permission (“contributor”)

In R1 or some later version, report level security will allow control of which reports and/or groups of reports (e.g., membership reports) are available to a user, as we have done with JReports. This involves:

* An additional table linking reports and user profiles, specifying report-level permissions.
* Additional functionality in the user profile UI to assign report permissions.

## 7. User launches new reporting portal application

Given that the report portal is enabled for the org and for that specific user, some new menu item or control will launch the portal:

* The portal will be in a separate browser window.
* The org key and user credentials will be provided at that time, probably in the URL.

Note that in the discussions in Dallas, what was proposed for an R1 release was to use ANet’s existing menu system. This could still be the R1 release model. The portal is included in this model to show the additional data and control flows necessary to implement it. Whether the reports are launched from ANet or the portal does not change the nature of the rest of the integration.

## 8. Portal provides UI to select report to execute, based on ANet user permissions

The portal will use web services to get the list of reports available to that user from the servlet, and display it’s report launch UI.

All filter selection is assumed to be implemented in the report itself, so the report will be launched immediately.

## 9. Portal calls the reporting service by URL to load the selected report into a window.

When the report is executed, it will be displayed in a different window from the portal.

* It is assumed that it is possible to do this via a URL, which must be able to specify:
* The report user (derived from the org information and the user role by a naming convention).
* The report to execute
* Whether this report session supports customization (unless this is implemented at the report user level)
* In order to prevent a URL from being reused to access data again outside of ANet’s security system, there must a system like Tableau’s trusted authentication to ensure one-ime execution of a report URL.
* Like ANet’s existing reports, report executions will be logged, although we won’t know how many times a report was then run in the interactive process, only how many times it was launched.

## 10. Report execution can scheduled within the report display window

Using a UI provided by the report server, be able to schedule execution of a report with the currently selected filter values.

## 11. Report accesses data from multi-tenant ADP database based on user

When the report executes, it must filter data for the correct tenant ID in the ADP, based on the org in the portal. Depending on the tool, these could be done in a number of ways:

* If the report server has good enough multi-tenant support, a report filter may be generated by the reporter server security system based on the user. (Jasper appears to have this capability).
* An org filter could be passed in from the portal to the report.

# Questions for vendors about integration capabilities

In order to develop ad hoc reports, is there a preliminary step to create a semantic model.

* What is the model called?
* What application is used to develop the model?

How can the model be exported as a file? What file type?

How is that exported model loaded into a different environment (for deployment)? (Automation).

When an ad hoc report is developed, how is it exported? What file type?

Do reports and models reside in something like folders? (Called what)

How is that exported report into a different environment (for deployment)? Control the target folder? (Automation).

Do the models also have to be imported, or are they built into reports already? Are they also in these folders? Is there automation support to deploy them?

Are permissions to reports in folders assigned to users, or is there also something like a user group or role? (Called what)

When assigning user or role permissions to a folder, what permissions can be set?

* Ability to run the report
* Ability to customize the report
* Ability to save the customized report

For deployment automation, is there an API or other means to create the following:

* A new folder for the org
* A new user (/role)
* Permissions for the user for the org folder and public folder.

Is there an API to allow an application to enumerate information on all reports available to a user, in a specific folder?

Is it possible to load a report into a window with a URL? If so, can the URL specify the following:

* A one-time use security token?
* The user? (Implied by the token?)
* Report parameters?
* Ability to modify the report
* Ability to save a modified report

When reports are scheduled for execution, is it from the report itself, with the same criteria?

How can we control which users can schedule execution? Either by URL, or by user permission?

Is it possible to use a report parameter to show or hide a filter configured in the report?

Is it possible to change column heading or filter names?

# R2+ Functionality

The following functionality may not be implemented in R1

* New report launch UI being designed by product
* Report level security.
* Automated setup of folders and users for orgs, and publishing of reports.
* Single site security. This will probably mainly be a function of workbook or datamart functionality, but may require some limited enabling in ActiveNet.