# “R1” model for integration of ActiveNet and Jasper Reports

## Assumptions

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

* The R1 integration is very simple, consisting of launching the Jasper repository browser in a tab, with a Jasper user whose permissions are such that only reports for that organization will be displayed.
* Permission administration will be implemented within ActiveNet. No report-level security will be implemented in R1.
* The Jasper report will be responsible for displaying any filter UI.
* Reports get their data from a multi-tenant ADP store.
* No AN user-level features in Jasper; e.g., no individual favorites.

## User experience

* The user will log into ActiveNet as usual.
* Based on their configured user profile, the user may have either of these permissions:
  + Jasper reader: The ability to execute Jasper and run reports for that org.
  + Jasper writer: The further ability to modify reports and save changes to the reports, or save-as to create new reports.
* If the user has the Jasper reader permission, a new menu item will be displayed.
* Clicking on that item will display the Jasper repository browser in a new tab.
* The user will only see reports available to that org; specifically:
  + The generic reports developed by Active and published during deployment.
  + Any customizations saved by staff of that organization.
* If the user has the Jasper writer permission, they can save any modifications they make to reports.
  + If the report is a generic report, they can only save-as to create a new report in the organization’s folder.

# Reference integration model



## Developers create domains, ad hoc views and reports

The report team will develop three types of objects in Jasper using Jasper Studio:

* Studio (“canned”) reports which cannot be modified by users. Filters will be built into these reports.
* “Domains”, which are semantic models to abstract the actual data model.
* “Ad hoc views”, which are starting points for ad hoc reports. Filters are also associated with these.

The reports, domains and views will be exported as files and checked into a repository, from which build images for deployment will be generated.

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

When a build image is deployed to a production environment, the deployment process must do the following operations against the report server in production, using the Jasper’s API:

* For each (new) org:
  + Create a Jasper organization.
  + Create a few generic users for that org for specific roles. (Currently reader and writer).
  + Create the permissions to allow these users to have access to appropriate folders.
* Publish the template reports to a generic folder shared by all orgs.

## Jasper Reports stores reports in folders

Jasper’s repository system system will be used to store reports in two types of folders:

* A “generic” folder containing the reports and views available to all orgs.
* One folder per organization containing its customized reports.

## Jasper Reports has a few generic users per org, granting folder access

There will be two generic users per org:

* The reader user will see in the repository and be able to execute reports in the generic folder and in the organization’s folder.
* The writer user will further be able to save to the organization’s folder (but not the generic folder).

There will a naming convention for these users based on the org key and user’s role.

## ActiveNet provides agency UI to control report permissions for ANet users

In R1, the permission administration is extremely simple:

* 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”)

## ActiveNet loads the Jasper repository browser into a window

In R1, ActiveNet will not provide a list of reports, or any report-level security. Instead, it will simply launch the Jasper repository browser in a window:

* If the org has the Jasper license and the user has the Jasper reader permission, a new menu item will be displayed.
* Clicking on that item will display the Jasper repository browser in a new tab.
* The repository browser will be automatically logged in as the correct user for that org having either reader or writer permissions.
* In order to prevent a URL from being reused to access data again outside of ANet’s security system, the report URL will contain an encrypted security token with an expiration time.

## 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.

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

When the report executes, it will filter data for the correct tenant ID in the ADP, based on the current Jasper user. This will be implemented in the domain for ad hoc reports, and for studio reports … ?

# Jasper Object Model

## Required outcomes

* The user for an org will only see reports available to that org; specifically:
  + The generic reports developed by Active and published during deployment.
  + Any customizations saved by staff of that organization.
* If the user only has Jasper reader permission, they can run reports but not save any modifications they make.
* If the user has the Jasper writer permission, they can save any modifications they make to reports.
  + If the report is a generic report, they can only save-as to create a new report in the organization’s folder.

## Naming convention

* Jasper object IDs are strings, and cannot be changed once the object is created. Because these IDs are case-sensitive, to eliminate confusion, all object IDs in the model below are all lower-case. (Which is the standard Jasper practice).
* To separate Jasper objects created for ActiveNet from those for other applications in the same Jasper instance, all Activenet object id’s and names will start with “an\_”.
* When created, Jasper objects have display names which are the same as the ID, but they can be changed. There is a naming convention for the display names below, but this is purely for readability, and the display name won’t effect any of the system operation described.

## Repository preparation

1. Delete all the subfolders under /Folder Template below. By doing so, when organizations are created, they will not have these automatically created subfolders, which aren’t necessary for the time being, and create additional confusion when users are saving reports.

(root)  
 /Organizations  
 /Folder Template  
 /Ad Hoc Components  
 /Temp  
 /Templates

Note: We don’t delete the Themes subfolder.

2. Under the public folder, create the following tree:

(root)  
 /public  
 /activenet  
 ad\_hoc\_views   
 data\_sources   
 domains   
 images  
 input\_controls  
 reports

What is shown above are the id’s; after creating the folder, change the display name to change underscores to spaces, and start each word with caps (e.g., so ad\_hoc\_views -> Ad Hoc Views).

## Roles

There are two root-level roles defined, which will be used in the pre-auth execution of reports:

(root)  
 /an\_reader (Ability to view reports in repository and execute them)  
 /an\_writer (Ability to saved modified versions of reports)

## Organizations

First, a Jasper organization must be created for an Anet org, as in this example:

(root)  
 /an\_1317 display name = an\_ymcala  
 /an\_1239 display name = an\_ymcasatx

Note the following:

* The organizations are created under the root, not under “organization”.
* The orgId is “an\_” followed by ActiveNet’s org\_id (an int).
* The org name is “an\_” followed by Activenet’s database name.

## Repository

As a result of the above, the repository will have the following structure:The following structure for reports is assumed:

(root)  
 /organizations  
 /an\_1239  
 /an\_1317  
 /public  
 /activenet  
 /Activities   
 /Domains  
 /Financials  
 /Memberships

The org folders under Organizations are auto-created, when an organization is created.

Note that some of these folders, like “public”, are displayed in mixed case, like “Public”, but the internal ID used in the API is lower case, which is what is shown here. To stay with the lower case convention, all folders we create, like “reports”, are also lower-case.

## Role-based repository permissions

The following permissions can be created once and will work for all organizations. In Jasper, if a permission setting is not overridden, the permission of the parent folder has effect. How show permissions are shown top-down, and in general, permissions are listed only if they differ from the permissions on the parent folder.

|  |  |  |
| --- | --- | --- |
| **Repository Path** | **Roles** | **Permissions** |
| /public | ROLE\_USER | No access |
| /public | an\_reader an\_writer | Read Only |
| /public/audit /public/diagnostic /public/monitoring /public/Samples /public/templates | an\_reader an\_writer | No Access |
| /public/adhoc /public/Resources | an\_reader an\_writer | Execute Only |
| /public/faith  /public/other applications | an\_reader an\_writer | No access |
| /public/activenet | an\_reader an\_writer | Read Only |
| /public/activenet/data\_sources /public/activenet/domains /public/activenet/images /public/activenet/input\_controls | an\_reader an\_writer | Execute Only |
| /organizations/ | an\_reader  ROLE\_USER | Read only |
| /organizations/ | an\_writer | Read + Write + Delete |

The following permissions must be assigned to each org’s subfolder after the org is created (this has been implemented as part of the SCM script which creates ActiveNet organizations using the Jasper REST API:

|  |  |  |
| --- | --- | --- |
| **Repository Path** | **Roles** | **Permissions** |
| /organizations/<orgId>/organizations /organizations/<orgId>/themes | an\_reader an\_writer | Execute |

Note that the same pattern will work for other applications like Faith One. Where the table above references an\_writer, it would reference fo\_writer. Where it references /public/activenet, it would reference /public/ActiveFaith.

## Org Users

Users for organizations will be created in two ways:

* Optional fixed “internal users” for testing
  + User with an\_reader role: userid, username and password = an\_<org>\_reader; ??? ROLE\_USER
  + User with an\_writer role: userid, username and password = an\_<org>\_writer
  + User with role\_superuser: userid, username and password = an\_<org>\_dev
* “External users” created through trusted authentication login
  + Userid = an\_<AN username>
  + Role = either an\_reader or an\_writer.
  + orgId=<orgId>

## User -based repository permissions for internal org users

For testing, any manually created users for an organization should be setup as in the following example:

|  |  |  |
| --- | --- | --- |
| **Organization** | **User** | **Role** |
| /organizations/an\_ymcala | an\_ymcala\_reader | an\_reader |
| /organizations/an\_ymcala | an\_ymcala\_writer | an\_writer |

## Access right model – test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Org\_Reader | Pass | Org\_Writer | Pass |
| Public report - view | Y | Y | Y | Y |
| Public report – execute | Y | Y | Y | Y |
| Public report – save | N | Y | N | Y |
| Public report – saveas | N | Y, Awkward | Only to same org folder | Y, Awkward |
| Public report – delete | N | Y | N | Y |
| Public report – edit properties (rename) | N | Y | N | Y |
| Same org report - view | Y | Y | Y | Y |
| Same org report – execute | Y | Y | Y | Y |
| Same org report – save | N | Y | Y | Y |
| Same org report – saveas | N | Y, Awkward | Only to same org folder | Y |
| Same org report – delete | N | Y | Y | Y |
| Same org report – edit properties (rename) | N | Y | Y | Y |
| Different org report - view | N | Y | N | Y |
| Reports in other paths | N | N | N | N |

# Jasper URL formats

## Server URL

In the documentation below, <server URL> is the base URL of Jasper, which for the Dev server is:

Dev: http://dev-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro

QA: <http://qa-jasperserver-vip.dev.activenetwork.com/jasperserver-pro>

Prod: <http://prod-jasperserver-vip.active.tan/jasperserver-pro>

Prod: https://activenetinsights.active.com/jasperserver-pro

## Login token

In the URL models below, there is a parameter pp (principal parameter):

&pp=<token>

This parameter causes Jasper to login as an “external user”, with a specific role and organization. Jasper creates an “external user” for that organization, assigns it the role, and logs in as that user. The token will be encrypted, to match a decryption class we provide for Jasper. Here is the contents of the token (example values and description):

u=an\_dchristie\_writer AN username, prefixed with an\_,   
 and postfixed with either \_reader or \_writer  
|r=an\_writer an\_reader or an\_writer, depending on whether the user should be able   
 to save any changes  
|o=an\_1317 orgId  
|exp=20150122150000-0800 expiration time of ticket; should be 3 min after now  
 Note UTC timezone adjustment based on org’s timezone

## Repository page launch URL

When an ActiveNet user runs Jasper (ACTIVENET Insights), they will be brought to the repository page, which shows the folders and reports available to that org/user. To launch the repository page for the right org and user, for a minimal implementation, use a URL in the following format:

<serverURL>/flow.html  
?\_flowId=searchFlow  
&sessionDecorator=yes Displays the browser with the top navigation frame  
&userTimezone= Etc/GMT+0800 Same timezone as in exp, but with sign reversed  
&pp=<token>

Sample reader URL:

http://qa-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/flow.html  
?\_flowId=searchFlow  
&sessionDecorator=yes  
&pp=u=an\_dchristie\_reader|r=an\_reader|o=an\_1317|exp=20160122160000-0600

# Build out for 15.3: Folder-level security

## Security model

ANE-27388 “Implement granular report security for ACTIVE Net Insights platform”, calls for “category-level” permissions for Jasper. Note that the Sharepoint document referenced by ticket also describes site-level security. Site-level security is outside the context of this document, and will have a completely different implementation model.

Per discussion with product, “category” means a Jasper repository folder. For the purpose of this security model, folders can be organized into these categories:

* Public folders: All organizations have access to a folder with URI “/public/activenet” (display name “/Public Folder/ACTIVE Net”). Within it, there are currently folders “Activities”, “Domains”, “Financials” and “Memberships”, although this list will grow as the Jasper implementation matures. The reporting team publishes reports and domains to these folders. Our organization users cannot modify them: they can’t modify reports in them, save new reports to them, create subfolders, etc. For these public folders, the folder-level permission will control whether a user with a specific user profile can view the folder and execute the reports in them.
* Organization root folder: Each organization has its own folder. For example, the organization an\_ymcala has a folder which appears to users a as path /, and display name an\_ymcala. Currently, all organization users can view everything in that folder, and users with “write” permission can save reports there, create subfolders, etc. In 15.3, a new user permission “Manage Custom Folder” will give users the ability to write to that folder, replacing the current write permission. This is primarily to limit what users can create organization subfolders.
* Top-level organization subfolders: Under the organization root folder, users with Manage Custom Folder permission can create subfolders. For example, they might create organization folders for Activities, Memberships, etc., mirroring the public folders, but the folder structure will be under the control of the organization. As with the public folders, the profile will grant a user permissions to that folder, either no access (can’t even see the folder), read-only access (can view the folder and execute reports in it), and write access (can save reports there, delete reports, rename reports, create subfolders).
* Lower-level organization subfolders: Because any user with write permission can create subfolders of a top-level org subfolder, no additional security will be applied to these subfolders. They will automatically inherit the permissions of the top-level folder

Here is an example:

|  |  |
| --- | --- |
| **Path** | **Access given** |
| \public\activenet | All users of organization with global Insights permission have read access |
| \public\activenet\activities | User profile controls whether a user can see this folder or not. Access is limited to read-only. |
| \ymcala | All users of organizations with global Insights permission have read access.  Users with “Manage Custom Folder” permission have write access. |
| \ymcala\activities | User profile controls whether a user has no access, read-only access, or write access |
| \ymcala\activities\special | Governed by profile access to \ymcala\activities |

## Integration model

Here is a high level description of the integration flow. Details of how the APIs and URLs are used are in later sections below.

* New permission for Manage Custom Folder.
* New table: There will be a new table in ActiveNet, Jasper\_Folders. Conceptually, this is similar to the AdhocReports table in the current JReports implementation. At a minimum, it needs to contain the URL (path) of the folder, and the label (display name) of the folder, a retired flag, and possibly an indicator of whether it’s a public folder or an organization folder (although this can be computed from the URI).
* Synchronize Jasper\_Folders: When the profile permission page is displayed, use the Jasper API to update Jasper\_Folders to match the current configuration. Note that Jasper id’s are immutable, including the folder URI, so the URI can be used to query Jasper\_Folders to see whether it is necessary to insert a new record or update the label. The sync process should also “retire” but not delete any records in Jasper\_Folders which are no longer in the source system, and clear the retired flag if the folder shows up again. In this way, we don’t destroy organization configuration data if there is a misdeployment of report folders.
* Permission UI: Display new sections of the profile permission page, showing the available public and custom folders, and allowing permission to be set, via a table Jasper\_Folder\_Profile\_Options similar to ProfileAdhocReportOptions in the JReports system.
* Set Jasper permissions for a user when launching Jasper: Rather than setting permissions at the time profiles are changed, it is much simpler and more reliable to set them when a user launches Jasper. However, there is an interesting problem. If we take the user we’re going to launch (e.g., an\_dchristie), we can’t assign permissions to that user, because the user might not exist yet; it will be created by the initial SSO. And SSO users cannot be created by the Jasper API; they can only be created by SSO. So instead, the permissions will be granted via a role, as follows:
  + Create a role corresponding to the user (e.g., a role an\_dchristie) if the role doesn’t currently exist.
  + Assign repository permissions to the role based on the user’s current profile(s), and the current Jasper\_Folder\_Profile\_Options settings.
  + In the encrypted pp token, eliminate the \_writer and \_reader suffixes to the user name (e.g., the user will be an\_dchristie, not an\_dchristie\_writer), and the role will be the same as the user name. The roles an\_writer and an\_reader will no longer be necessary.

## General considerations

### REST API Security

The API calls to enumerate the folders, create the role, and assign permissions to it, must also be performed as some user. Just as when we launch Jasper today, we can provide an encrypted pp parameter as part of the API query string. However, to separate this user from the organization users who login, we will use a different user id in a pp like this:

u=an\_activenetservlet Special user for issuing API calls  
|r=ROLE\_ADMINISTRATOR  
|o=an\_1317 orgId  
|exp=20150122150000-0800 expiration time of ticket; should be 3 min after now  
 Note UTC timezone adjustment based on org’s timezone

### Case-sensitivity

Note that Jasper’s ids are case-sensitive. The convention in the integration is that any objects we create have lower-case id’s. This will eliminate any confusion. However, when we enumerate through folders that humans have created (not the automation), they may have mixed case, so the ids have to be stored and used exactly as fetched by the API.

### State of testing

The documentation below is based on the Jasper documentation, and tests of GET methods. The PUT or POST operations have not been tested yet, so the documentation represents the current understanding.

### Other design considerations

* Everything having to do with Jasper interface, including the REST API calls and the launch process, should be in the JasperSoft class.

## Enumerate through folders for an org

This uses Jasper’s REST API to generate an XML listing of folders available to the current organization:

<serverURL>/rest\_v2/resources  
?type=folder  
&pp=<token>

**Test URL:**

If you’re already logged into Jasper, you can exercise this API with the following URL (no encrypted pp is necessary):

http://dev-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/rest\_v2/resources?type=folder

**Report data returned**

The data returned looks like this:

<resources>

<resourceLookup>

<creationDate>2015-05-19T22:28:55</creationDate>

<description/>

<label>ACTIVE Net</label>

<permissionMask>2</permissionMask>

<updateDate>2015-05-29T15:44:36</updateDate>

<uri>/public/activenet</uri>

<version>1</version>

<resourceType>folder</resourceType>

</resourceLookup>

<resourceLookup>

<creationDate>2015-05-19T22:28:58</creationDate>

<description/>

<label>Activities</label>

<permissionMask>2</permissionMask>

<updateDate>2015-02-04T00:18:34</updateDate>

<uri>/public/activenet/activities</uri>

<version>0</version>

<resourceType>folder</resourceType>

</resourceLookup>

<resourceLookup>

<creationDate>2015-06-26T08:04:44</creationDate>

<description/>

<label>Anita</label>

<permissionMask>2</permissionMask>

<updateDate>2015-06-26T08:04:44</updateDate>

<uri>/Anita</uri>

<version>0</version>

<resourceType>folder</resourceType>

</resourceLookup>

</resources>

When parsing this data:

* Ignore folders with URI’s starting with /themes or /organizations. These are internal folders created by Jasper itself, which we hide from the users
* For public folders: We want to show only immediate subfolders under /public/activenet. So in the example above, we will capture the URI /public/activenet/activities with the display name Activities.
* For organization folders, we will capture any immediate subfolder under /, other than public, themes or organizations. So in the example above, we will capture the URI /Anita with the display name Anita.

## Determine whether there is already a role for a user

To determine whether the role an\_<username> exists, do a GET to this API:

<serverURL>/rest\_v2/organizations/orgID/roles  
?search=an\_<username>  
&pp=<token>

This should return XML like this:

<roles>

<role>

<externallyDefined>false</externallyDefined>

<name>an\_dchristie</name>

<tenantId>an\_1317</tenantId>

</role>

</roles>

If there is no role for the user, the code will need to create one.

Note that this API seems to do a partial string match, so if you query for an\_dchr, you’ll find the user for an\_dchristie. This means it’s possible you’ll get more than one record, or get a record for the wrong person whose username starts with the current username. So you will have to look through the entire XML returned for an exact match.

## Create a role for the user

If the role does not exist, create one, by doing a PUT to the URL:

<serverURL>/rest\_v2/organizations/orgID/roles/an\_<username>  
?pp=<token>

The payload should be an empty role descriptor:

<role>

</role>

## Set permissions for the user

POST to this URL:

<serverURL>/rest\_v2/permissions  
?pp=<token>

The payload is a JSON string like this:

{ “permission” :[

{ “uri”:”/<folder uri>”,

“recipient”:”role:/an\_<username>”,

“mask”:”0” }

{ “uri”:”/<folder uri>”,

“recipient”:”role:/an\_<username>”,

“mask”:”2”

},

{ “uri”:”/<folder uri>”,

“recipient”:”role:/an\_<username>”,

“mask”:”30” }

]}

The mask value for each folder is as follows:

0 = no access (can’t view folder)  
2 = read only (can view folder contents and execute reports)  
30 = read-write-delete (can also save reports there, rename reports, delete reports, and create subfolders).

Here is an example of how the permissions should be set:

|  |  |
| --- | --- |
| **URI** | **Mask** |
| /public/activenet | 2 for all users |
| /public/activenet/activities | 0 if user has no permission 2 if user has permission |
| / | 30 if user has Manage Custom Folder permission 2 otherwise |
| /organizations | 0 |
| /themes | 0 |
| /Anita | 0 if user has no permission 2 if user has read permission 30 if user as write permission |

Note that it is necessary to write data for all public and organization top-level folder URIs, not just ones the user have access to. Otherwise, if folder access is removed from a profile, it would remain in Jasper.

## Launch Jasper

The basic URL structure for launching Jasper in unchanged:

<serverURL>/flow.html  
?\_flowId=searchFlow  
&sessionDecorator=yes Displays the browser with the top navigation frame  
&userTimezone= Etc/GMT+0800 Same timezone as in exp, but with sign reversed  
&pp=<token>

However, as noted above, the actual URL to launch Jasper for the current user will be changed, in that the username and role in the encrypted pp token will be different:

u=an\_dchristie AN username, prefixed with an\_   
|r=an\_dchristie Same as username  
|o=an\_1317 orgId  
|exp=20150122150000-0800 expiration time of ticket; should be 3 min after now  
 Note UTC timezone adjustment based on org’s timezone

## Repository browser launch URL

To launch the repository browser to show all the reports available to the organization, for a minimal implementation, use a URL in the following format:

<serverURL>/flow.html  
?\_flowId=searchFlow  
&sessionDecorator=no Displays the browser without the top navigation frame  
&pp=<token>

**Test URL for reader:**

Note that these test URLs have sessionDecorator turned on for testing, so we can see the logged in user, and log out easily.

http://qa-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/flow.html  
?\_flowId=searchFlow  
&sessionDecorator=yes  
&pp=u=an\_dchristie\_reader|r=an\_reader|o=an\_1317|exp=20160122160000-0600

**Test URL for writer:**

http://qa-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/flow.html  
?\_flowId=searchFlow  
&sessionDecorator=yes  
&pp=u=an\_dchristie\_writer|r=an\_writer|o=an\_1317|exp=20160122160000-0600

## Alternate report listing

The following URL format allows us to show a list of all the reports available to the organization, without the folder structure. As such, it mixes together the shared reports and the custom reports, but if the org uses a consistent naming convention, they can tell them apart:

<serverURL>/flow.html  
?\_flowId=searchFlow  
&mode=search  
&filterId=resourceTypeFilter  
&filterOption=resourceTypeFilter-reports  
&pp=<token>

Sample for reader user

<http://qa-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/flow.html>  
?\_flowId=searchFlow  
&mode=search  
&filterId=resourceTypeFilter  
&filterOption=resourceTypeFilter-reports  
&sessionDecorator=yes  
&pp=u=an\_dchristie\_reader|r=an\_reader|o=an\_1317|exp=20160122160000-0600

## Report launch URL

[<serverURL>/flow.html](http://qa-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/flow.html)  
?\_flowId=viewReportFlow  
&reportUnit=<report uri>  
&sessionDecorator=no   
&pp=<token>

**Test URL for reader:**

## <http://qa-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/flow.html?_flowId=viewReportFlow&reportUnit=/public/activenet/reports/NewMembersDrillDown&sessionDecorator=yes&pp=u=an_dchristie_reader|r=an_reader|o=an_1317|exp=20160122160000-0600>

**Test URL for writer:**

## <http://qa-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/flow.html?_flowId=viewReportFlow&reportUnit=/public/activenet/reports/NewMembersDrillDown&sessionDecorator=yes&pp=u=an_dchristie_writer|r=an_writer|o=an_1317|exp=20160122160000-0600>

## Enumerate through reports for an org

This uses Jasper’s REST API to generate an XML listing of reports available to the current organization:

<serverURL>/rest\_v2/resources  
?type=reportUnit  
&folderUri=/  
&pp=<token>

**Test URL:**

Note that by convention, a special external user an\_activenetservlet is used for this call, to distinguish it from actual ANet users.

[http://qa-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/rest\_v2/resources?type=reportUnit&folderUri=/&pp=u=an\_activenetservlet |r=|o=an\_1317|exp=20160122160000-0600](http://qa-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/rest_v2/resources?type=reportUnit&folderUri=/&pp=u=an_activenetservlet%20|r=|o=an_1317|exp=20160122160000-0600)

**Report data returned**

The data returned looks like this:

<resources>

<resourceLookup>

<creationDate>2015-01-22T15:30:12</creationDate>

<description/>

<label>New Members Drill Down - Public</label>

<permissionMask>2</permissionMask>

<updateDate>2015-01-22T20:43:42</updateDate>

<uri>/public/activenet/reports/NewMembersDrillDown</uri>

<version>1</version>

<resourceType>reportUnit</resourceType>

</resourceLookup>

<resourceLookup>

<creationDate>2015-01-22T21:27:43</creationDate>

<description/>

<label>New Members Drill Down - YMCASATX</label>

<permissionMask>2</permissionMask>

<updateDate>2015-01-22T21:27:59</updateDate>

<uri>/NewMembersDrillDown</uri>

<version>1</version>

<resourceType>reportUnit</resourceType>

</resourceLookup>

</resources>

When parsing this data:

* Label is displayed in the menu as the report name.
* Uri is used when launching the report, as the “reportUnit”.
* If we want to separate generic reports from the org’s customized versions, we can tell from the path in the uri. Generic reports will be in the path /public/activenet/reports; org-specific reports are in the root /. (This is a byproduct of this method logging in as an org user)

## Jasper logout URL

There are a few issues around logout.

If a

<serverURL>/exituser.html

# Jasper customizations:

There are several customizations to our Jasper implementation, which are critical to the SSO model shown above. To understand these, it’s necessary to first understand the default operation of the pp ticket:

* Jasper creates a user for the specified org, with the name specified, if it doesn’t exist, like “an\_dchristie\_reader”.
* This is flagged as an “external user”. This means that the user has no password, and can’t be used for a manual login, but only as part of a token. So if we want to grant org users manual login rights, we’ll have to manually create users for them.
* Jasper then assigns any roles specified to the user. However, the roles in the token by default are “external roles”, and have to be further mapped to internal roles.
* The roles are additive. So if you login with an\_dchristie with an\_reader, then an\_dchristie has the an\_reader role. If you later login to the same user with an\_writer, that user will have both the an\_reader and an\_writer roles. This means it can never be a reader again. For this reason, the samples above have two users, an\_dchristie\_reader and an\_dchristie\_writer.

Our Jasper implementation has the following custom configurations, which are critical for its operation:

**Token based login:** This setting is necessary in order for the pp token to work at all. Settings in a config file specify the format of the token, and what class to use to decrypt the token. Currently, the decryption class is a standard “plain text decryption” class Jasper provides, so we can test with plain text tokens. We will develop our own decryption class and corresponding encryption code in ANet as part of the integration.

**Map by name:** This customization means we don’t have to map the roles. When we provide an\_reader in the token, it will automatically map that to the internal role an\_reader.

**Map to root:** With map by name, Jasper will map the role in the token to a org-specific role of the same name. This means we have to create both roles each time we create an org, and assign more permissions. With this setting, Jasper will map the role by name to a root-level role, so one role can server for all organizations.

The following customizations also need to be done, with help from Jasper:

**Overwrite roles**: With this setting, each time a login token is processed, the existing roles will be cleared, so the user will have only the currently specified roles.

**Disable save button if no write permission:** Currently, if a user does not have write permission, the save button still shows up, and is enabled. If you click it, you will be brought to the save-as dialog, and can navigate the repository tree, but the save button won’t be enabled on any of the folders. Jasper said there is a setting to hide the button if there’s no write permission. Even better would be product’s request that the button be visible, but disabled.

# Product questions

* Permissions for scheduling reports

# Jasper Questions

## Feature requests

* Disable save button if no write permission.
* Replace roles on mapping, rather than merging.

## Report browser control

* Control whether user can schedule or not?

## Deployment

* Automation of org setup
* Automation of report publishing

## “Localization”

* How is a “locale” determined?
* If a locale file is updated, when does Jasper read it?

# Post R1 functionality

## Enumerate through reports for an org

**Get all reports for org**

<serverURL>/rest\_v2/resources  
?type=reportUnit  
&folderUri=/  
&pp=<token>

<http://qa-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/rest_v2/resources?type=reportUnit&folderUri=/&pp=u=an_servlet|r=an_reader|o=an_1317|exp=20160122160000-0600>

**Report data returned**

The data returned looks like this. We would save the <label> and the <uri>.

<resources>

<resourceLookup>

<creationDate>2014-12-05T21:23:09</creationDate>

<description/>

<label>Membership Usage - public</label>

<permissionMask>1</permissionMask>

<updateDate>2014-12-05T21:24:35</updateDate>

<uri>/public/reports/MembershipUsage</uri>

<version>2</version>

<resourceType>reportUnit</resourceType>

</resourceLookup>

</resources>

## Create all objects for an org

## Publishing a report

# Reference documents

All docs: <http://community.jaspersoft.com/documentation?version=12873>

REST API: <http://community-static.jaspersoft.com/sites/default/files/docs/jasperreports-server-web-services-guide_4.pdf>

HTTP API: <http://community.jaspersoft.com/system/files/restricted-docs/js-ultimate-guide-5.6.pdf>

Authentication by ticket: <http://community.jaspersoft.com/system/files/restricted-docs/jasperreportsserver-auth-cookbook_0.pdf>

# Saved notes

**Get generic reports**

<serverURL>/rest\_v2/resources  
?type=reportUnit  
&folderUri=/public/activenet/reports  
&pp=<token>

http://qa-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/rest\_v2/resources?type=reportUnit&folderUri=/public/activenet/reports&pp=u=an\_servlet |r=an\_reader|o=an\_1317|exp=20160122160000-0600

**Get org-custom reports for org**

The following works, but without the (should be unnecessary) “&recursive=false “, it returns all reports for ymcala, plus all reports in /public:

<serverURL>/rest\_v2/resources  
?type=reportUnit  
&folderUri=/  
&recursive=false  
&pp=<token>

http://qa-adp-jspserv-01w.dev.activenetwork.com:8080/jasperserver-pro/rest\_v2/resources?type=reportUnit&folderUri=/&recursive=false&pp=u=an\_servlet|r=an\_reader|o=an\_1317|exp=20160122160000-0600