Name of Standard: **NRS Standards for Jasper Reports**

Standard Custodian Name: Fary Eriksson

Submitter Name: Jeff Johnson

Stakeholder(s) Impacted by Change: **BPMs, Technical Architects**

Type of Change: **Low**

Change to Standard: **Added information on NRSRS (ISSS instance of Jasper). Updated broken links. Added info on export format issues, and scheduled reports being deleted during migrations. Updated info on using the P_REPORT_DIR parameter. Added reference to MS SQL Server support in JCRS.**

Standard

Version Control

Date	Author	Version
June 7, 2017	Jeff Johnson	1.2.0

1. Introduction
   This document defines standards for design, development, delivery, and deployment of *Jasper Reports*. Jasper Reports is the preferred technology for sector applications to use to provide preconfigured, parameterized reports.

2. List of any SDLC Deliverables this standard directly relates to :
   Perhaps the **Application Delivery Checklist** and **Software Design Description**

3. **Standard**
   (Attached)

Supporting Documentation: Jasper documentation, some references to Jasper community support articles

Corporate Services for the Natural Resource Sector

Information Management Branch

Standards for Jasper Reports

Last Updated:	**June 5, 2017**
Version:	1.2.0
Document:	NRS_Standards_for_Jasper_Reports

Table of Contents

1. Version Control

Document Version	Revision	Date	Author(s)	Change Reference
1.0.0		March 3, 2014	Jeff Johnson	Initial release
1.1.0		June 12, 2016	Jeff Johnson	Minor updates, plus information on load-balancing and WebADE secure by organization feature.
1.2.0		June 5, 2017	Jeff Johnson	Minor updates, plus references to NRSRS

2. Introduction

2.1 Purpose

This document defines standards for design, development, delivery, and deployment of *Jasper Reports*. Jasper Reports is the preferred technology for sector applications to use to provide preconfigured, parameterized reports.

2.2 Audience

This document is directed at vendors and sector staff who will be creating or managing development of Jasper Reports for the Sector.

2.3 Scope/Exclusions

The scope of this document covers development of all Jasper Reports delivered to or maintained by the Sector.

Where conflicts, if any, are perceived between this document and other standards, the Business Portfolio Manager must be consulted.

2.4 Assumptions

It is assumed that the audience has working knowledge of the current development and delivery standards for applications built for the Sector, and some technical familiarity with web development, the Java language, Jasper Reports, Oracle databases, and SQL.

2.5 Definitions

The following definitions apply throughout this document.

2.5.1 Standards

A standard is a specific statement of the rules and constraints governing the naming, contents, and operations of software. A standard must be followed. There is a contractual obligation on the part of the vendor/developer to adhere to all relevant standards.

2.5.2 Guidelines

A guideline is a method or custom, which through common usage has become an accepted method of work. A guideline is not enforced, and is not a standard.

2.5.3 Sector

Unless otherwise specified, "Sector" is taken to collectively mean the Ministries and agencies which are included under the umbrella of the Natural Resource sector.

2.5.4 CSNR

Corporate Services for the Natural Resource sector, the common Corporate Services Division servicing all the ministries and agencies in the Natural Resource sector. Information Management Branch (IMB) is part of CSNR, and has the mandate to formulate and maintain application development standards.

2.6 Contacts

Most inquiries regarding these standards should be directed to the Business Portfolio Manager that is assigned to the project being developed. Technical queries and suggestions for improvements may be sent to the CSNR Application Architecture Services positional account, CSNR.Application.Architecture.Services@gov.bc.ca.

3 Overview

Many sector applications use preconfigured, parameterized reports to provide business information. Jasper Reports deployed on Jasper Server Community Edition is the preferred technology solution for sector applications which include these reports. It is a free, open-source product, easily deployed on the Sector Middle Tier (SMT) infrastructure and scalable by adding more servers. It is a proven product widely used in industry, and accessible to our developer community. It is also customizable via Spring Security to use the sector's WebADE security API.

More detailed information on choosing a technology platform for parameterized reports is included in the NRS Standard for Formal Reports.

3.1 Sector Jasper Server instances

The sector has two implementations of Jasper Server. NRSRS, the Natural Resources Sector Reporting System, is the ISSS instance, and connects to the ISSS WebADE. It is used for ISSS applications only. JCRS, the Jasper Corporate Reporting System, connects to the Forests WebADE, and is used for all non-ISSS applications.

These two instances currently use different versions of Jasper Server, and have significantly different delivery methods. An upgrade to JCRS planned for fiscal 2017 will eliminate or minimize these differences.

NRSRS-specific guidelines and documentation are maintained separately. When this document and the NRSRS documentation disagree, the NRSRS documentation should be considered the authoritative source.

3.2 Technologies

Jasper Reports includes several separate components – the Jasper Reports library (which can also be used standalone, as part of an application), the report designer (iReport and Jaspersoft Studio), and the report server (Jasper Reports Server Community Edition). The sector also uses Apache Tomcat as the application server, as described in the Sector Middle Tier documentation. MySQL is used to host the Jasper Reports Server repository.

3.2.1 Jasper Reports library

JCRS currently deploys Jasper Reports library version 5.0 on its servers. Developers of non-ISSS applications should use this version of the reporting library, whether deploying on Jasper Reports Server or used standalone.

NRSRS deploys Jasper Reports library version 6.3. Developers of ISSS applications should use this version of the reporting library

3.2.2 iReport Designer and Jaspersoft Studio

iReport is a report designer for Jasper Reports. It includes and uses Jasper Reports library. iReport Designer version 5.0.0 contains the version of the reporting library deployed on the sector's servers, and should be used to develop reports for non-ISSS applications.

Jaspersoft Studio, an Eclipse plugin, is required to develop reports for NRSRS. The currently preferred version is 6.3.0.

3.2.3 Jasper Server

JCRS currently deploys Jasper Reports Server Community Edition version 5.0. Developers of non-ISSS applications should use this version of the report server.

NRSRS currently deploys Jasper Reports Server Community Edition version 6.3. Developers of ISSS applications should use this version.

3.2.4 Oracle

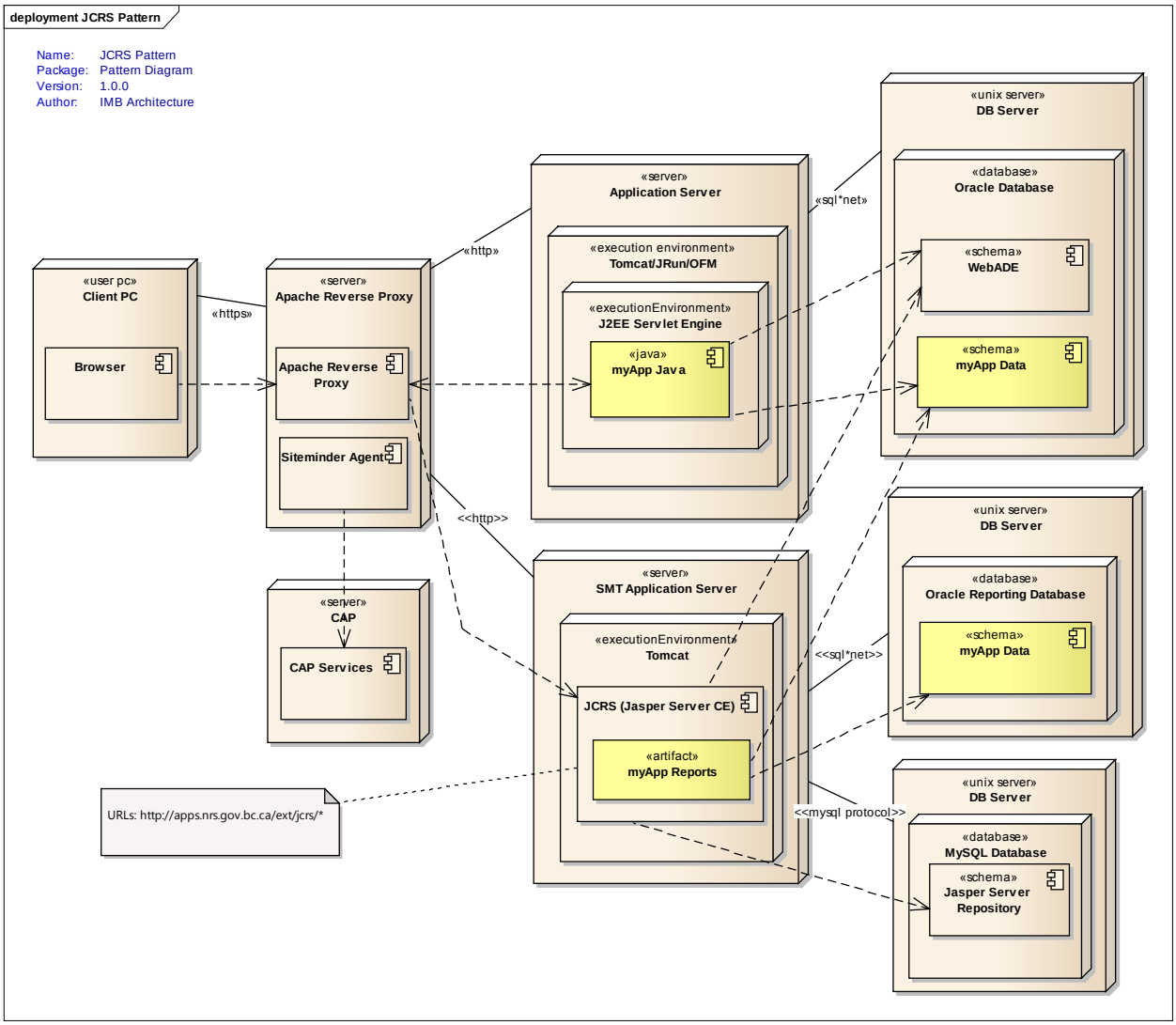
Most sector data is stored in Oracle databases. The sector currently uses Oracle 12c.

3.2.5 Microsoft SQL Server

Some sector data resides on Microsoft SQL Server. JCRS only includes a JDBC driver for SQL Server. NRSRS does not.

3.3 Application Pattern with JCRS

This pattern shows a typical WebADE application using reports deployed on JCRS. The tan colour highlights code, data, and reports belonging to the application. Other scenarios are possible.



3.4 Axioms

The following principles guide report development in the sector:

3.4.1 A Report should deal with presentation only

The code of a report should only deal with the presentation of a dataset. The dataset itself should be sourced from a stored procedure in a sector database in most cases. Any parameters **must** be validated and processed in the stored procedure. Only a report that does a simple select from a single view or table with no parameters can do so directly without a stored procedure.

3.4.2 Reports are authorized by WebADE

Access to reports should be authorized via the WebADE security API. Our Jasper Reports Server infrastructure includes WebADE integration code so that the WebADE can provide the roles assigned to the user, and these roles are used to grant access to reports. ADAM is used to assign roles to authentication profiles within a JCRS application, and delegate the ability to manage user authorizations to application administrators.

Applications that have Jasper reports in the application itself, not on a server, need to have them secured by WebADE roles as well.

3.4.3 Reports should be hosted on a JCRS report server

In general, reports should be hosted on the sector's JCRS or NRSRS platforms. They provide an alternate path for users to access reports, and the ability to schedule reports, email reports, and run them in the background. It also allows the sector the ability to centrally manage reporting infrastructure, and collect reporting metadata.

Reports can be embedded in an application when necessary. This might be required for apps with unique requirements not met by JCRS/NRSRS. Some applications were also delivered with embedded reports during the period before JCRS and NRSRS were available.

3.4.4 Reports should be web-friendly

Most reports will be run interactively via the browser. This imposes several restrictions: users expect a response quickly, and the response must be delivered within the limits of the various session timeouts in the sector's operating environment.

No report available interactively should take longer than 5 minutes to run to conclusion. Ideally, the report should run in 1 minutes or less. The source query and default values of input controls should be written to ensure this.

Longer reports can be run non-interactively, via Jasper Server's scheduling capability.

4 Jasper Server Repository

Each Jasper server will support many reporting applications. The following standards help ensure that reports can be easily deployed and delivered for all applications on a server.

Developers must configure their own server to comply with these standards.

4.1 Folder layout

JCRS uses a standard folder layout designed to simplify user navigation of the folder hierarchy within the Jasper Server web interface, support common components, and to ease management and migration of applications by separating each application's objects.

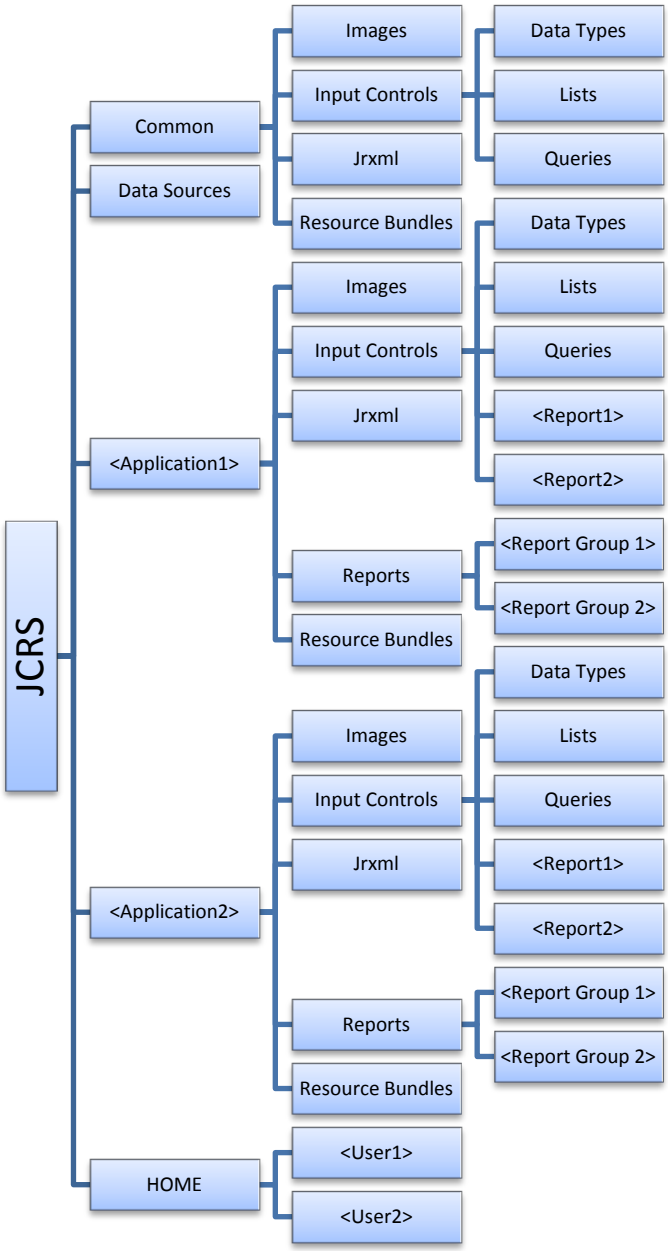


Figure 1 - JCRS Folder Layout

4.1.1 NRSRS Layout

NRSRS uses a very similar folder structure, except the root folder is called NRSRS, there is currently no Common folder, and data sources are stored in a subfolder named for the application acronym.

4.2 Folder Descriptions

The intended purpose of each folder in the sample folder layout is described below.

4.2.1 JCRS Root folder

This folder separates JCRS folders from more 'generic' Jasper Server objects. All JCRS objects will be in subfolders of this root folder.

4.2.2 Common folder

This folder contains shared objects available to all reports. These are managed by the JCRS application administrator. An external link to the Common folder is included in every application's Subversion repository, allowing developers to download the contents. Currently, there are no resources available here, but these are planned.

4.2.3 Data Source folder

This folder contains data sources for each reporting application. As part of the initial setup of a reporting application, the developers and JCRS application administrator will agree on a name for the data source(s) used by the application. By default, data sources will be named *JCRS_<acronym>* or *JCRS_<acronym>_<role>*.

Because a data source contains user and password information, the sector copy of the Data Sources folder is not available for developers to download.

4.2.4 Application folders

Each application will be self-contained, and store all of its own objects in its folder. Each application folder will use the application acronym as its name. A template will be created when a reporting application is initially set up. The template will be available for developers to download. Developers will store exports of their application folder in their Subversion repository.

4.2.5 Resource folders

Each type of resource supported by Jasper (Images, Input Controls, Data Types, Lists, Queries, Jrxml, and Resource bundles) will have its own folder in both the Common and Application folders. Since Data Types, Lists, and Queries are only used for Input controls, they were made subfolders of the Input Controls folder.

'Jrxml' is used for shared subreports, or other shared JRXML. The main report code should be a local resource of the report definition in most instances. An exception would be, for instance, if two reports have the same main report code but different input controls.

4.2.6 Report folders

Every application stores its reports in its own Reports folder. Applications that deploy numerous reports can create subfolders to organize them.

The Reports folder is the only folder that is visible to the users navigating the Jasper Server UI. **All reports, and only reports are placed in an application's Reports folder and its subfolders.** This ensures that users navigating the Jasper Server UI will see just the reports they have available to them, and none of the supporting resource files. Report-specific Input Controls should be locally-defined (bundled with the report) or placed in a subfolder of the Input Controls folder named for the report.

4.2.7 User Personal Folders

Every IDIR and BCeID JCRS and NRSRS user will have a personal folder created in the HOME folder, such as

/JCRS/HOME/idir_<username>

The default security permissions will provide write access to only the specified user. This folder is the only one most users will have write access to.

4.3 Jasper Security

4.3.1 Users

Most users of JCRS and NRSRS will be 'external' users, with BCEID and IDIR accounts.

Reporting applications which wish to have their reports made available for programmatic access by code running outside a web browser (Java code running on a server, for instance) can optionally have local Jasper accounts created for this purpose. These accounts will be named 'LOCAL_<APP>'.

The server administrator account(s) are also local Jasper accounts.

4.3.2 Jasper Roles

Two key roles are used to secure most objects:

ROLE_WEBADE_JASPER_USER allows access to the root folder and the Common folder. All ADAM profiles created for JCRS and NRSRS include this role; it is required in order to use the Jasper Server GUI.

ROLE_WEBADE_JCRS_<APP> roles are per-application roles, which allow access to the application reports and objects, including the application's Data Source(s). A typical reporting application will have only one role – either users can run the reports or not. For instance, an application named FOO will have just a ROLE_WEBADE_JCRS_FOO role.

Applications which have multiple user populations, each of which must be granted access to different sets of reports, may need more than one role. During the initial setup of each reporting application, these roles must be specified and created.

Application BAR, which shows some reports to all users and others to only payroll clerks, might have roles ROLE_WEBADE_JCRS_BAR_USERS and ROLE_WEBADE_JCRS_BAR_PAYROLL.

ROLE_LOCAL_JCRS_USER is the analogue of ROLE_WEBADE_JASPER_USER, and is granted to the local accounts created for programmatic access to reports. ROLE_LOCAL_JCRS_<APP> roles are also created for apps which want to allow programmatic access.

NRSRS has identical setup, except NRSRS is used instead whenever a name would use JCRS. For instance, the AS application on NRSRS is secured by ROLE_LOCAL_NRSRS_AS.

There is a special role, ROLE_RUN_AS_ADMINISTRATOR, used by some of the WebADE integration code. The HOME user folder is secured with this, allowing the user's folder to be created on login even though the user does not have permission to write to the /JCRS/HOME directory.

The administrative accounts used to manage the server are also secured by local roles.

4.3.3 WebADE Integration

In order to provide WebADE roles, JCRS and NRSRS use a custom library and configuration, integrating with Jasper Server via the Spring Security API.

Developers do not need to use this, but can emulate it with appropriately-named local roles.

Developers can use this WebADE integration if desired, and it is available on request. Most developers will need to use the WebADE Developer module rather than Siteminder.

Some WebADE profiles and their granted roles can be 'secured by organization', indicating the user is granted the role when acting on behalf of a certain organization. Within JCRS/NRSRS, a user may only act for one organization at a time. When a user is authorized to use profile(s) secured by more than one organization, JCRS/NRSRS will prompt the user for which organization to use, and will also populate a set of built-in parameters which indicate the user's chosen organization, which can be used in reports. These are described later.

Two special roles, ROLE_WEBADE_VIRTUAL_HAS_MULTIPLE_ORG_ROLES and ROLE_WEBADE_VIRTUAL_HAS_ORG_SELECTED are also granted in the appropriate circumstances, and can be used to secure folders (you could have a set of reports that rely on the users selected organization, secured by ROLE_WEBADE_VIRTUAL_HAS_ORG_SELECTED, for instance).

Secure by organization support cannot be easily emulated in a development environment without the WebADE integration code, however.

4.3.4 Siteminder

All Jasper server URLs are currently secured by a Siteminder agent. Only IDIR and BCeID users can access JCRS; NRSRS is currently available only to IDIR users.

Public access to JCRS is being considered for the future. Projects wishing to host publically available reports must contact CSNR Application Architecture early in order to determine if it can be provided. Significant lead time will be required.

4.3.5 Jasper permissions

The following permission scheme is used to ensure that users navigating the Jasper Server GUI only see the reports accessible to them, and do not see (but have access to) any resources used by their reports.

Except for the Data Sources folder in JCRS only, all permissions are applied to folders, and objects in folders inherit permissions.

ROLE_WEBADE_JASPER_USER and ROLE_LOCAL_JCRS_USER have Read permission in the root folder. This is overridden and replaced with Execute only permission in the Common folder and Data Sources folder. These roles allow all users to navigate the JCRS folder hierarchy.

The individual data sources in the JCRS Data Sources folder (and the per-application subfolders of the Data Sources folder in NRSRS) have permissions overridden so that only the application's role(s) have Execute Only permission, and all other roles have no permission.

The Individual user folders within the HOME folder have permissions overridden so that only the individual user has Read/Write/Delete permission, and all other roles have no permission.

Application folders have permissions overridden at the root level so that only the application roles have Read permission. In all the resource folders, this is overridden so that the application roles have only Execute permission.

The report folder normally inherits permissions, so the application role(s) have Read permission. This is sufficient for most applications. Applications which want reports divided by role must specify their preferred scheme when creating or modifying their reporting application.

5 Report Invocation/Delivery

Reports can be executed in several ways on Jasper Server.

5.1 Interactive report delivery methods

5.1.1 Jasper server UI

It is possible to define in Jasper Server rich parameter entry screens, with features such as SQL driven select lists, static select lists, calendar widgets, cascading dropdowns, etc. These are usually accessed by logging on to the Jasper Server interface and navigating to the appropriate reports.

After entering the parameters, and running the report, the report can be viewed page by page in the browser, or it can be exported to a variety of formats.

5.1.2 URL link with or without decoration

Instead of searching for the reports in the Jasper Server interface, the report can be accessed via URLs such as this one:

https://testapps.nrs.gov.bc.ca/ext/jcrs/flow.html?_flowId=viewReportFlow&reportUnit=/JCRS/JCRS_Demo/Reports/FTA_Tenure_Status_Reports/FRTR100___District_Tenure_Status_Summary&decorate=no

The link can be customized to run the report directly, to decorate the browser window with the corporate banner, etc. The report parameters can be embedded in the link, or they can be selected later. The user will be authenticated and authorized before being granted access to the report.

5.1.3 REST Web Service call

The Jasper REST API calls are documented in the Jasper Reports Server Web Services Guide. REST web services can easily be used by applications, via the browser or by server-side code. REST web services that use GET differ from running reports via links because they will not create a user interface. An unauthenticated REST request will not be redirected to a login screen, instead an "Authentication Required" HTTP Status will be sent.

Example:

https://testapps.nrs.gov.bc.ca/ext/jcrs/rest_v2/reports/JCRS/JCRS_Demo/Reports/FTA_Tenure_Status_Reports/FRTR100___District_Tenure_Status_Summary.pdf?p_district_number=34&p_inc_file_type=A01&p_exc_file_type=*

runs the report with the specified input control values and exports it as PDF. The ‘pdf’ extension can be replaced with html, csv or any other supported output format extension.

In the sector environment, our WebADE integration allows JavaScript running in a Siteminder-authenticated browser session to make REST calls seamlessly, without needing to use the Jasper REST login first.

REST calls in any other context (for instance, by server-side Java code) need to use the Jasper REST login method to create a session. Applications that use this capability will also need to have a local Jasper account with appropriate permissions created.

When application code makes the REST login API call, it ***must*** save the JSESSIONID and ROUTEID0 cookies that are set in the response, and include them in all subsequent requests that session. These two cookies are used to ensure requests go to the server that is maintaining your

session in the CSNR environment (which has multiple clustered Tomcat servers). Requests that do not save and set these two cookies may go to a different server, and will fail.

Note that when using the REST API, you may encounter a caching issue when you :

- Request a report with a certain set of parameters, then
- Independently, take an action that causes the information accessed by the report's data source to be changed, then
- In the same session, request the report again with the same parameters

To avoid this, either make all report requests in a new session, or else follow the steps in the linked article in the appendix.

5.1.4 WebADE Reporting Extension

Some sector applications that originally used Crystal reports were developed to use the WebADE Reporting Extension, a Java library, to process and schedule report requests. The reporting extension uses the Jasper REST API to communicate with the Jasper server.

This is a legacy component. Newly developed applications should use URL links and/or the Jasper REST API directly.

5.1.5 SOAP Web Services

Jasper server also implements parallel SOAP Web Services with mostly the same functionality as the REST Web Services. The REST API is preferred.

5.1.6 Timeout issues

There are several timeout periods involved in the sector infrastructure. Reports must be designed to complete within a limited time frame to accommodate this. No report available interactively should take longer than 5 minutes to run to conclusion. Ideally, the report should run in 1 minute or less. The source query and default values of input controls should be written to ensure this.

Most of the Jasper Server UI uses Ajax. It is possible for a user's session to be timed out, at which point the Ajax elements of the interface become non-responsive. A page refresh is necessary to make the Jasper Server UI responsive again.

5.2 Offline report delivery

Jasper Server supports scheduled tasks that deliver reports at set times and optionally repeated at a certain set interval. Reports must be delivered to folder where the user has write access. An email with an attached report or link to one can optionally be sent as part of a scheduled task.

The only folder available by default is the user's personal folder. JCRS and NRSRS have some customization to ensure this folder will be the default folder for output when creating scheduled tasks.

5.2.1 Email Delivery

Email delivery of reports run through the Jasper Server UI requires creating a scheduled task.

However, applications should email reports themselves whenever these reports are being created on a one-time basis. This reduces the proliferation of report output in the users' personal folders. They would need to retrieve the report via the REST API, then handle emailing it themselves.

Applications which need to create recurring scheduled reports can use the REST jobs API to create scheduled tasks that deliver reports via email.

5.2.2 FTP

In later releases of Jasper Server than that currently used in JCRS, but available in NRSRS, scheduled reports can be delivered via FTP. JCRS may enable this option in the future, but it is currently available only in NRSRS.

6 Report Design

Reports created for JCRS must comply with the following standards.

6.1 Development Tools

Developers require an installation of Jasper Server Community Edition. They must use the same version currently used for JCRS/NRSRS, as listed in section 3.2.3 of this document.

Developers may use any development tool they wish to develop reports. However, it is strongly recommended that they use the release of iReport or Jaspersoft Studio that includes the same version of the Jasper Reports library currently deployed in JCRS/NRSRS, as listed in section 3.2.2 of this document.

iReport development has ceased in favour of Jaspersoft Studio, an Eclipse plugin. Jaspersoft Studio is required to develop for NRSRS, and will be the recommended report designer for JCRS as well when the JCRS servers are updated with a newer version of Jasper Server. This change is planned for fiscal 2017.

6.1.1 Additional libraries

Developers will require the JDBC driver for Oracle (or any other databases their reports must retrieve data from). This must be added to both the server and development tool.

The Jasper server API and metadata API libraries should be added to the iReport and/or Jaspersoft Studio classpath. These needed to be copied from Jasper Server. In Jasper Server CE 5, the two files are:
jasperserver-api-common-5.0.0.jar
jasperserver-api-metadata-5.0.0.jar

The PL/SQL executor must also be added to the Jasper server to support Oracle stored procedures. This is a DLL provided with iReport. Instructions are provided on the Jasper Community website.

If your reports use any of the fonts in the Microsoft Core Fonts for the Web package, you will need a font extension library JAR. One is available on request.

6.2 Common Resources export

An export of the Common resources will be provided. At inception, the Common resources folder is empty. Objects suitable for shared use will be developed in the future, and accumulated from releases of reporting applications.

6.3 Layout

Letter sized reports are preferred (612x792 or 792x612 pixels). To start a new report, use the Blank Letter or Blank Letter Landscape report templates from iReports.

Margins should be 20 pixels on all sides.

All fields must be aligned horizontally, vertically and should either exactly touch each other or there should be a space between them. For example, if a field starts at left=20 and the width is 40, the next field must start at 60 or at 70 or even better at 80, not at 61. This is because excel

columns will be created for each value of the coordinates. Having a cell that ends at 60 while the next one starts at 61, would result in a 1 pixel wide column which is undesirable. Alignment is made easy by the "snap" feature of the designer, or the developer can watch the field sizes / coordinates and make sure they are multiples of round numbers.

Alignment considerations apply to both horizontal and vertical coordinates.

6.4 Design for Reuse and Ease of Maintenance

Reports should isolate common, changeable elements so they need only be changed in one place. Formatting should be done with styles wherever possible. Changeable strings (organization names, for instance) should be placed in string resources. Shared components used by multiple reports in a project should be identified and only a single copy kept in the appropriate resource folder.

6.5 Branding

Once they become available, one of the subreports from the common area should be included in the title section band of the report. These subreports include the report name, ministry name and other identifying data for the report. The header subreport should have no margin, and the width of the subreport should be the width of the parent report less the parent report margins. The information displayed in the report header may be passed in via parameters, or retrieved from the database.

6.6 Subreports

The JRXML files of subreports should be either locally defined in the report unit, in the Common Jrxml resources folder (/JCRS/Common/Jrxml), or in the application Jrxml resources folder (/JCRS/APP/Jrxml). Subreports used by just one report can be included in the report, or in a subfolder of the application Jrxml folder (/JCRS/APP/Jrxml/ReportID). They should initially be referenced as repo:subReportName, that is, so they create indirect reference names.

Indirect references are placeholder names that must be manually linked to the resource when uploading the JasperReport. The syntax for an indirect reference contains only a placeholder name for the resource. When uploading the report, the server will prompt you to locate and link the subreport.

This establishes a link between the subreport and the parent, and moving/renaming the subreport will not break the link. You will be unable to delete the subreport before deleting all referencing reports, however.

The same applies to images held in the repository or to other resource files. They should be linked during upload.

6.6.1 Subreport expressions

A drawback to using this way of addressing subreports, is that reports have to be modified when they are edited in the report designer, because they will not find their subreports using repo: addressing. While developing, it is possible to use a subreport expression instead.

Create 2 parameters, $P{SUBREPORT_DIR} and $P{SUBREPORT_EXT}.

For SUBREPORT_DIR, set the parameter expression with this:
($P{REPORT_CONTEXT} != null) ? "repo:" : ""

Note: the built-in parameter : $P{REPORT_CONTEXT} is null when report is executed from IReport but contain a value when report is executed from the Jasper Server. This way, it set $P{SUBREPORT_DIR} to "repo:" when executed on the server and set to the same directory as the main report when executed from IReport

Similarly, for SUBREPORT_EXT, the same logic is applied. Set the parameter expression with:
($P{REPORT_CONTEXT} != null) ? ".jrxml" : ".jasper"

Finally, for all subreport expressions use the following:

($P{SUBREPORT_DIR} + "report_name" + $P{SUBREPORT_EXT})

where report_name is the filename of your report without extension.

You won't be prompted for local resources (at least for the subreports) when you upload your report, so you will also have to upload your subreport JRXML files as resources with the name being their filename, with extension (eg "subreport_001.jrxml").

6.7 Naming Conventions

6.7.1 Jrxml Files

These should follow the naming convention REPORT ID - Report Name.

Example: FRTR100-District Tenure Status Analysis

(The actual file name would have the .jrxml extension; the resource in the Jasper server will not.)

6.7.2 Report Descriptions

One of the common ways users will access reports on the server is the Library view of the Jasper Server GUI. They will then be able to narrow down the list of reports displayed by doing a search. This search is a pure text search (with no wildcard or regex search) against the name and description of the report. It does not include the names of folders in the report's URI path.

To support user searching, appropriate strings should be included in the description of the report. This should include the application acronym and the names of any folders used to separate reports into categories. For instance, if the FTA application had a set of reports in a Tenure Reports subfolder, the descriptions of the report should include 'FTA' and 'Tenure'.

6.7.3 Resource Ids

Any resource stored in the Jasper Server repository has an ID that cannot be edited after creation. In some tools, it is not possible to specify the id and the system defaults it to the name, with all spaces replaced with underscores.

Note: do not have resources with Ids that differ only in case. This will cause difficulties with imports and exports (which may be done on case-insensitive file systems).

If it is possible to specify the ID, it should be entered as the name, with spaces or other invalid characters replaced by underscores

Example: FRTR100_District_Tenure_Status_Analysis

6.7.4 Parameters

Parameter names should always be lowercase, to distinguish them from the built-in parameters.

Words should be separated by underscores.

To maintain consistency with the current CRS, parameter names should start with 'p_'. Multi-select parameters can use ‘pm_’.

Example p_district_no

6.7.5 Queries

Queries used to populate parameters in an input control should be named ‘q_<parameter_name>’.

For example, a query used to populate a dropdown list for an input control affiliated with a parameter called ‘p_district_no’ should be named ‘q_district_no’.

6.8 Standard Report Parameters

Every report should include a set of standard parameters, after the built in parameters. Built in parameters with special names are always in uppercase. The way they are populated may change depending on what report metadata exists in the business database.

Note that parameters in Jasper reports are case sensitive.

ORACLE_REF_CURSOR
Type: java.sql.ResultSet This parameter is only used for reports that are driven by stored procedures. This parameter is automatically added by the editor when the query language is changed to plsql. If it is not added, you have to add it manually and bind the REF CURSOR parameter returned by the stored procedure to it. Do ***not*** check “Use as a prompt” for this parameter.
LoggedInUsername
Type: string This built-in parameter is populated automatically by Jasper Server. It should be used to identify the user when the user is accessing the report via the Jasper Server UI. When an application makes a REST call to run a report on behalf of a user, the username will have to be passed as a regular parameter. Do ***not*** check “Use as a prompt” for this parameter.
p_report_id
Type: java.lang.String The report ID should be passed in by jasper server via a read only parameter. Do ***not*** check “Use as a prompt” for this parameter. Example: FRTR100

p_report_category
Type: java.lang.String (optional, can also be retrieved from metadata) The report category should be a default value, or should be retrieved from metadata in a common header subreport. Do ***not*** check “Use as a prompt” for this parameter.
p_report_name
Type: java.lang.String (optional, can also be retrieved from metadata) The report name should be a default value, or should be retrieved from metadata in a common header subreport. Do ***not*** check “Use as a prompt” for this parameter.
p_business_parameter_1 - business parameter 1 **p_business_parameter_2 - business parameter 2** **etc**
Parameters that are used as input parameters for a stored procedure should be named p_<proc parameter name>

6.8.1 Secure by Organization Parameters

If the user has access to one or more WebADE roles secured by organization, JCRS will populate an additional set of parameters.

WebAdeOrganizationId **WebAdeOrganizationName** **WebAdeOrganizationDescription** **WebAdeOrganizationCode** **WebAdeOrganizationType**
Type: java.lang.String These parameters identify the user’s selected organization, if any, when they have one or more roles secured by organization.
WebAdeAllOrganizationIds **WebAdeAllOrganizationNames**
Type: java.util.Collection These parameters contain all of the user’s organizations if they have one or more roles secured by organization, regardless of whether they have an organization selected.

6.8.2 Security Note about lexical parameters

Jasper server allows the use of lexical parameters that use the $P!{paramName} syntax, as opposed to the $P{paramName} syntax. This is a powerful feature which should be used only when the security implications are clear. Lexical parameters create the opportunity for SQL injection.

Always using bind parameters (the usual $P{p_user}) with a stored procedure should prevent SQL injection attacks.

6.9 Containing references

References to other resources should never go outside the top level JCRS/NRSRS folder.

Most resources should be stored in the application folder.

The only resources outside the application folder that can be accessed are the application's data source(s) in the Data Sources folder, and common resources in the JCRS/Common folder.

6.10 Charts

Charts should be developed using only the libraries available on Jasper Server Community Edition (JFreeChart). Chart types delivered only with Jasper Server Pro are not available on the JCRS servers.

6.11 Mapping

The Google Maps component included with Jasper Server and iReport is not suitable for sector use. The mapping framework currently being developed for sector applications as part of the Integrated Sector Strategic Solution (ISSS) project will address mapping for Jasper Reports.

6.12 Fonts

Fonts used in reports need to be both available on the server platform and the user's platform (for most output types) or be licensed for embedding in PDF reports. This limits the supported fonts.

JCRS currently supports only the fonts provided with Jasper Server (DejaVu Sans, DejaVu Sans Mono, and DejaVu Serif) as well as those included in the Microsoft Core Fonts for the Web package (Andale Mono, Arial, Comic Sans, Courier New, Georgia, Impact, Times New Roman, Trebuchet, Verdana, Webdings).

The DejaVu set of fonts are recommended. The recommended font for the body text of most reports is DejaVu Sans - Size 8.

A link for a font extension JAR file containing the Microsoft Core Fonts for the Web is available in the Appendix.

Use of other fonts will be considered; however, additional time will be required to confirm licensing terms and create and deploy a font extension on our servers.

6.13 Scripting

The scripting language used in expressions must be Java (not JavaScript, groovy, beanshell, etc). While we recognize that a strong argument can be made in favor of a scripting language, we feel that portability across versions of the software is most important. Scripting languages behaviour tends to change considerably between versions. In addition, the version of groovy distributed with iReports is significantly different from the version installed on the Jasper Server, making testing difficult.

Encapsulating behaviour in Java scriptlets is not necessary in most cases. When scriptlets are used, thorough justification must be presented

6.14 Reports Unsuitable for HTML Display

Some reports may be meant only for display in certain formats, and may be unsuited to display in HTML format (or even impossible to display, causing some browsers to crash). We have identified a possible workaround for these reports. This must be customized for every format, however, which requires additional time to develop and deploy on our servers.

7 Administrative Tasks

This section specifically refers to JCRS. Setup and delivery tasks for NRSRS are different, and maintained separately.

There are several administrative tasks involved in creating and maintaining a reporting application for JCRS.

7.1 Creating a New JCRS Reporting Application

Several tasks need to be performed to initially set up a reporting application in JCRS. This will require a minimum of two weeks advance notice. The following information is required:

Application name and acronym

These are used to create the application folder and data source(s) on the JCRS servers.

Database proxy accounts for report data sources

The data source(s) used by an application require proxy accounts to connect to the database. Creating these and managing their database security permissions is the responsibility of the application administrator.

WebADE security scheme

Most reporting applications are expected to use a simple security scheme with just one role, which allows access to all the application's reports. Applications with more complex security needs must have these presented and reviewed.

The JCRS administrator must create the JCRS WebADE role(s) and profiles for the new reporting application and delegate permission to authorize these profiles to the people designated by that application`s administrator.

When your application is ready for use, you will be informed of the paths to your data source(s) on the JCRS servers and a folder template will be created in your Subversion repository. The template will have permissions set on it for your application roles.

7.2 Code Delivery

CSNR Application Deliveries will be preparing more detailed standards for how to deliver Jasper reports code with your application. These will be available separately.

Current delivery requirements are summarized below.

Jasper reports for deployment on the JCRS servers should be in a jasper-server folder in your Subversion repository.

That folder will be:

https://a100.gov.bc.ca/pub/svn/<appName>/trunk/source/jasper-server/src/main

where <appName> is your application acronym.

There are three subdirectories in each application's Subversion repository. 2 are under sector control, and one is for your application code.

One folder will contain your application's Jasper code:

https://a100.gov.bc.ca/pub/svn/<appName>/trunk/source/jasper-server/src/main/resources

One folder contains an export of the JCRS Common folder:

https://a100.gov.bc.ca/pub/svn/<appName>/trunk/source/jasper-server/src/main/common

The other folder contains an export of the Jasper template for your project: – a bare directory tree with permissions set, and your data source(s) with placeholder values:

https://a100.gov.bc.ca/pub/svn/<appName>/trunk/source/jasper-server/src/main/template

When you deliver your application's Jasper code, you should deliver an export of your application folder from your development server **without permissions and without dependencies**. The js-export utility (found in the buildomatic folder of the Jasper distribution) should be used for this. A sample command line might be

```
js-export.sh  --uris  /JCRS/Appname --output-dir  /path/to/project/folder/source/jasper-server/src/main/resources
```

This export will include dependencies (such as resources in the Common folder, and the application's data sources). Dependencies outside the application's folder **must** not be included in your SVN repository. They can either be deleted, or marked as ignore in SVN.

Developers can do this manually, or a perl script which does an export using the Jasper REST APIs and then keeps only certain specified folders can be provided on request.

Jasper reports embedded in an application should be delivered with the other web source files in the application.

7.2.1 Important Note on Case Sensitivity

Although resources inside the Jasper folder structure (and inside an export ZIP file) are case-sensitive, exports of the code will at times exist on case-insensitive file systems (such as Windows NTFS).

Jasper exports do not currently correct for case-sensitivity differences between the server and the export file system. They will fail silently when exporting, and the exported code will fail when subsequently imported.

It is important and necessary that you do not have any resources with URIs that differ only by character case. It is safest to adopt a consistent case for all your resource URIs to avoid this issue.

7.2.2 Important Note on Export Format Consistency

The js-export command can export either to a folder, or to a ZIP file. The Jasper GUI can export only to a ZIP file. Whichever method you use, you should consistently use the same export format. Exports to a folder fix up some special characters to ensure filenames are legal (such as underscores), and flag Jasper server to do a correction on import. Exports to a ZIP file do not perform such fixups. If you switch between export formats, this can cause your export to have confusing history in Subversion or Git.

7.3 Importing Existing Jasper Code for JCRS

The artifact created by a build in Jenkins of a project with Jasper code will be a jcrs-reports.zip file which contains both _template and main project subfolders (unmodified, that is, with

template values for the data source). Both of these folders can be used as source directories to import code from an existing project.

You can also manually create an import folder:

Make a working copy of the trunk\source\jasper-server\src\main\resources folder in a new location.

Add a copy of the trunk\source\jasper-server\src\main\template\resources\JCRS\Data_Sources folder to the resources\JCRS subfolder in the working copy

Your working copy of the resources folder will now work as an import folder for js_import.

7.4 Migration

When you request a migration, the basic procedure is:

- The current application folder in JCRS is deleted.
- The template folder for the application (with data source placeholder properties fixed up) is imported.
- The application's Jasper code (with dependencies added) is imported.

7.4.1 Important Note on Jasper Scheduled Tasks During Migrations

Because the entire application folder is deleted, any scheduled tasks created using the application's report will also be deleted and will need to be recreated.

Appendix – Useful Resources

The following resources may be useful for developers of Jasper reports for the sector.

Jasper Community Site http://community.jaspersoft.com/

The iReport Ultimate Guide and Jasper Reports Server Ultimate Guide available here are free and especially recommended. The Jasper Reports Server Web Services Guide is also necessary; it publishes the REST and SOAP web services interfaces.

Article describing how to use Oracle stored procedures in Jasper

Article describing a caching issue when using the REST API

Jasper Server Community Edition downloads http://sourceforge.net/projects/jasperserver/

iReport downloads http://sourceforge.net/projects/ireport/?source=directory

Older versions of Jasper Server CE were removed in May 2014 due to licensing issues with the MySQL JDBC driver. The ministry maintains copies of the currently-supported versions of iReport and Jasper Server.

- iReport 5.0 Installer
- Jasper Server 5.0 CE Installer for Windows 64-bit
- Jasper Server 5.0 CE WAR file distribution

Font Extension for Microsoft Core Fonts for the Web MS Core Fonts for the Web JAR file

CSNR Application Deliveries email: CSNRApplicationDelivery@gov.bc.ca

NRS Standards Inventory: NRS Standards Inventory

CSNR Technical Architecture email: CSNR.Application.Architecture.Services@gov.bc.ca

Test JCRS URL: https://testapps.nrs.gov.bc.ca/ext/jcrs/

Production JCRS URL: https://apps.nrs.gov.bc.ca/ext/jcrs/

Integration NRSRS URL: https://dlvrapps.nrs.gov.bc.ca/int/nrsrs

Test NRSRS URL: https://testapps.nrs.gov.bc.ca/int/nrsrs

Production NRSRS URL: https://apps.nrs.gov.bc.ca/int/nrsrs

NRSRS Deployment Guidelines on Confluence