pmsquare Journal
simplifying complexity

A Rough Guide to CMS

By Paul Mendelson

May 2016

Introduction

As the demand for information grows, so does the demand for availability. Instead of separate platforms, users expect their reports integrated directly into applications. Modern web sites can present data in a much flashier way than native Cognos, but web or application developers are rarely skilled Business Analytics experts. While it is easy to connect to a standard RDBMS, how can these developers pull data from a cube? Additionally, there are many graphing engines available, but these often require specialized knowledge. Cognos graphs can be delivered directly as an image.

Cognos Mashup Services allows these web or applications developers an easy way to retrieve data from existing Cognos reports. This puts the burden of building the reports and queries on the Cognos experts. When the developer needs to retrieve this data, he or she could simply retrieve the output and render it directly on the page.

There are a number of ways that Cognos can deliver the outputs. Traditionally Cognos reports are delivered in HTML through the report viewer. In CMS there are a number of additional methods.

1. HTML and HTML Fragment will provide the output, with all the styling, as a fully formed HTML. Any graphs can be embedded in this output in base64. This is probably the easiest format to work with, as the developer simply needs to embed the output directly onto the page. The biggest problem with this HTML, when coupled with styling, can become extremely large.
2. LDX. Layout Data XML is an XML format that describes the entire output.
3. Dataset. A simple XML output. There is no formatting information.
4. DataSet Atom/JSON. The XML output converted to ATOM or JSON. DatasetJSON is the output developers seem to prefer.
5. Image. Not so useful for a multipage monster of a report, but there may be times when you want to deliver the output in this format. For example, if you're printing the output, it may be easier to simply return an image than to worry about how different readers would render the same HTML.

When a request is sent to Cognos, the gateway will redirect the request to the dispatcher as usual. All of the normal Cognos processing occurs at this point, however the report is rendered in an XML format. The report is then processed by the presentation service and converted into the format you select.

The request itself is formed as a normal URL. Per the documentation a REST request is low overhead compared to a SOAP request, and will work better with larger volumes of data. For the purpose of this document all examples will be given as a REST GET request.

**Anatomy of a CMS call:

```
http://SERVER/ibmcognos/cgi-bin/cognos.cgi/rds/OUTPUT-
TYPE/SOURCE(storeID or searchPath)(/optional
outputFormat)?options=value&option2=value
```

The first part should look familiar, it's a normal call to the gateway. As with regular report calls, Cognos requires the authentication cookies in place. So whatever application is being built needs to have a way of authenticating the user and setting those cookies. One way might be to embed the credentials in the parameters. My current client built a custom authentication provider, and will check for a predefined cookie.

```
/RDS
This instructs the application that we are using the
reportDataService.
(http://myCognosServer/ibmcognos/cgi-bin/cognos.cgi/rds)
```

/OUTPUT-TYPE

The most common value I've used here is reportData and outputFormat. reportData will run the report in the outputs listed above, while outputFormat allows you to export the report to PDF or Excel. You could also use this to logon, log off, retrieve a list of prompts, or even run the prompt page. There are plenty of other options that I've personally never used, but that an application developer might need. ATOM will allow you to retrieve all of the details of the report (owner, contact email), or even a quick generated thumbnail without data.

```
(http://myCognosServer/ibmcognos/cgi-
bin/cognos.cgi/rds/reportData)
```

/SOURCE

The location of the report can be encoded as the storeID, the searchPath, path (a simplified version of the searchPath), or a conversationID. Using the searchPath may be easier in the long run, as the storeID can be changed when overwriting the report with a new version. The conversationID is used when using a secondary request, such as performing a drill down action or running a report asynchronously.

```
(http://myCognosServer/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/searchPath/content/folder[@name=%60CMS%20Example%60]/folder[@name=%60Reports%60]/report[@name=%60myList%60])
```

/outputFormat

If we were using the outputFormat, we could specify an output format here.

```
(http://myCognosServer/ibmcognos/cgi-bin/cognos.cgi/rds/outputFormat/searchPath/content/folder[@name=%60CMS%20Example%60]/folder[@name=%60Reports%60]/report[@name=%60myList%60]/spreadsheetML)
```

?OPTIONS

There are far too many options available for this short document. Needless to say, almost every setting you can think of can be set in the options here. I'll list the ones I most often use.

&async

```
(auto/manual/off)
```

A synchronous request (async=off) will maintain an open connection to the client until the output is ready and fully delivered. This is the easiest to use when developing, but in a production environment with a large user base the sys admins may through a fit.

Auto will send a redirect to a URL which will periodically poll Cognos for the output.
Manual will send the URL without the redirect.

Which is used is down to the application developer. In both cases a connection is NOT maintained, and the sys admins will have fewer reasons to be angry.

&drillthroughurls

(true/false)

A drillthrough in the Cognos Report will become a span with the definition embedded inside. The application developer can pull those out and use them in secondary requests or even as a direct link to the other report.

&embedImages

`(true/false)`

Graphs are normally saved as images in the content store and retrieved when called. Using this option will convert that image into Base64 and embed it directly into the report.

&fmt

`(layoutDataXML, HTML, HTMLFragment, JSON, Image, DataSet, DataSetAtom, DataSetJSON)`

Very important. This is how you define the output format. If this isn't defined the output will be LDX.

&includeLayout

`(true/false)`

By default Cognos will return the output sans tables or divs. When expecting the output in HTML, this will make things look very strange.

&inlineStyles

`(true/false)`

HTML only. For dashboards this won't have much of an impact, but if you have a large list this is indispensable. By default Cognos will put all of the styles inline. I've had one report containing a table with 3,000 rows and ~50 columns reduce in size from ~80MB to ~13MB.

&prompt

`(true/false)`

If you have any optional parameters, make sure to set this to false or you might get an unpopulated parameter error.

&p_parameter

Any parameters you have defined in your report can be set here.

&selection

This will call a specific named element in the report. You can use this to call, for example, a specific graph or list. A note of warning - when using this to call a div, table, or singleton, make sure to have includeLayout=true.

Real Life Example

While the preceding has barely scratched the surface of what we can do with CMS, the capabilities provided are extensive. We could build an entire application powered by CMS. In fact, at my current client, that's exactly what we're doing.

The data source is a Dynamic Cube based on Exadata. The reports are, for the most part, heavily styled. Pixel perfect type reports. Almost every report can be broken into two sections, summary and detail. The portal is built by web developers using Bootstrap. And the entire structure, what is visible, the values of the prompts, even the report links themselves, are derived from Cognos reports.

We have two types of reports. First we have the auxiliary reports. These are the reports that ultimately power the portal. Reports available to that user, prompt values, portal settings, and other details are pulled from simple list reports in DataSetJSON format. The designed reports are called using &fmt=HTMLFragment&selection=Detail (or Summary). The portal will paste the contents of those reports into the report section of the portal.

By using Bootstrap we are given a level of interactivity generally not found in normal Cognos reports. In some list reports we have a "MORE" link that will open a hidden drawer beneath the row. What's happening in the background is Cognos is actually running a separate report (via CMS) and the portal is pasting the results into the new row. To the end user the experience is completely seamless.

In other cases we have clickable question marks to give context to certain items. In the background we have an HTML item that pulls the ID for that item from the query, and the portal is running an auxiliary report to get the description for that field.

Even the structure of the portal may appear differently for different users. An auxiliary report is run on first login, describing exactly what the user is permitted to see.

For more information about this system, and how PMsquare can help you implement it, please contact us today at Contact@pmsquare.com.