Rational Quality Manager's REST API: Tips, Tricks, and Best Practices

Michael Freeman, michael.freeman@lpsvcs.com / mlfreeman@gmail.com
Sr. Applications Development Analyst, Lender Processing Services

QM-1845B

Innovate2013
The IBM Technical Summit

Stay ahead.

IMPORTANT: Legal Disclaimer Instructions for IBMers

- IBMers must include the next slide (verbatim) after your title slide.
- IBMers must also include the mandatory “Acknowledgements and Disclaimers” slide (see slide 10) at the end of your presentation before the closing “Thank You” slide.
  - You will need to customize the “Acknowledgements and Disclaimers” text in red appropriately.

Please note the following

How to use this template

- To allow all masters of your presentation to be updated correctly, download this template to your hard drive and copy your existing slides into the new template
- See slide notes for further formatting instructions
- See below for suggested color palette. Core colors are blue and green. Additional accent colors are purple, teal and orange, if needed.

🟩	🟩	🟩	🟦	🟦	🟦	🟦
Green 1	**Green 2**	**Green 3**	**Blue 1**	**Blue 2**	**Blue 5**	**Blue 6**
R140	R23	R0	R131	R0	R0	R0
G198	G175	G138	G209	G176	G100	G63
B63	B75	B82	B245	B218	B157	B105

🟪	🟦	🟧
Purple 2	**Teal 1**	**Orange 2**
R127	R0	R221
G28	G166	G115
B125	B160	B28

About the Speakers

- Michael is a Senior Applications Development Analyst with Lender Processing Services.
- He has been working with Rational Functional Tester since 2007 and Rational Quality Manager since 2010.
- He has had to interface with RQM from systems written in both Java and C#.

REST – What is it?

- REST: Short for REpresentational State Transfer
- Requests and responses are built around the transfer of representations of resources.
  - –A "resource" is typically just a unique URL.
  - –The "representation" is data either received from that URL or sent to that URL.
  - –Most of the time the data is an XML document.
- Typically performed over HTTP, but doesn't have to be.

REST – What is it?

- Stateless:
  - –Unlike a website, no client context is stored on the server between requests.
  - –Each request from any client contains all of the information necessary to service the request, and any session state is held in the client.
- 4 Main Actions: GET, PUT, POST, and DELETE.
  - –These were borrowed from the HTTP protocol.
- Typically, the HTTP action changes rather than the URL

REST vs Plain HTTP

- For example, let's imagine that we have a user database that is manageable over HTTP.
- Using the non-restful approach it would probably look something like this:

  /user_create

  /user?id=123

  /user_edit?id=123

  /user_delete?id=123

- A restful application would probably expose an API with a single base resource.

  /user

  /user/123

REST vs Plain HTTP

- If you want to retrieve the user with ID 123 you send a GET request to /user/123.
- If you want to create a new user you simply send a POST request to that URL with the data you want to create.
- If you want to delete the user, you send a DELETE request to /user/123.
- If you want to update the user with ID 123 you simply send the update using a PUT request to /user/123.
  - On some APIs you have to GET the resource first, update the fields, and send the updated resource back.
  - On other APIs you can just send the fields to update.

REST vs SOAP

- REST is *NOT* SOAP.
- They are two completely different design approaches, each with pros and cons.
- REST centers around Create/Read/Update/Delete of objects, identified by their URLs.
- REST is a technique/set of principles, so REST APIs can differ slightly.
- SOAP is an actual set of documented specifications – if you don’t implement the specifications you won’t be able to communicate with anyone else.
- For more, start with Wikipedia: https://en.wikipedia.org/wiki/Representational_state_transfer

REST vs SOAP

- SOAP APIs can expose arbitrary actions, not just create/read/update/delete.
  - Can be seen as similar to a function library/DLL/JAR in an application.
  - However, SOAP is more complicated to work with, so REST APIs are appearing more and more.
- Imagine an invoicing system and imagine that you wanted to find overdue invoices.
  - A SOAP API could expose a *getOverdueInvoices()* method that you could call.
  - A REST API would in most cases require the client to "know" what makes an invoice overdue.
    - Either the REST API would expose something akin to a WHERE clause or the client would be responsible for doing WHERE clause style filtering.

Why use RQM REST API?

1. Integration with your custom applications/tools.

– If you have a testing tool for a specific niche (e.g. hardware), you could write something that parses its output then uses RQM's REST API to upload the results to RQM.

2. Customized data importing/exporting.

- When we set up RQM at LPS, we had ~5,000-7,000 testcases across 8 apps we needed to load from Excel files into RQM.
  - Every project had done something slightly different with what was originally the same template (e.g. added/removed columns, added/removed sheets in the file, etc).
  - Every project had different ideas on what they wanted to save from their Excel files into RQM.
    - One project wanted me to set categories on every testcase based on the file (and sometimes other information in the file).

Why use RQM REST API?

- (continued)
  - Another project wanted me to detect certain words/strings and link to a RQM keyword rather than upload the original Excel step.
  - Some projects (but not all) had TFS defect numbers in the Excel files that they wanted copied to RQM.
  - IBM's default RQM importer for Excel files was nowhere near flexible enough, so I was tasked with writing importers for each project, so I turned to the RQM REST API.
  - I made one prototype RQM importer program in Java that used Apache POI to read Excel files one row at a time and Apache HTTPClient to send the data to RQM.
  - Once the prototype was approved, I made customized copies for each team (in order to handle their unique requirements).
  - It saved us many hours when compared to hand importing any of this stuff.

Why use RQM REST API?

3. Automation, Automation, Automation

- Many basic (and more complex) tasks may be automated with the REST interface:
  - Testcase Creation
  - User Management
  - Execution Record Management
- Some REST API Limitations
  - Lab Manager Interface is not adequate
  - Can be very slow with high-latency networks or large operations

About the API: Overview

- Main Page for Documentation: https://jazz.net/wiki/bin/view/Main/RqmApi
- Basic Usage: https://jazz.net/wiki/bin/view/Main/RqmApi#Usage_Basics
- 8 Main Actions:

HTTP Method	**Purpose**
GET	Read a resource or feed of resources.
PUT	Create or update a resource with a known ID (see Content-Location header).
POST	Create a resource and generate a unique ID (see Content-Location header).
DELETE	Delete or hide the resource from the Rational Quality Manager. Note, resources are never completely deleted for auditing purposes.
LOCK	Lock a resource rendering it read-only.
UNLOCK	Unlock a resource rendering it write-able.
SNAPSHOT	Create a resource snapshot (see Content-Location header).
PROPFIND	Read a list of resource snapshots.

About the API: Login Process

- You have to simulate a form posting to j_security_check under your server (e.g. https://rqm.example.com/qm/j_security_check)
- You have to supply the user name and password as fields named j_username and j_password.
- The response body can be ignored. To tell if you logged in OK, look for a HTTP header named “X-com-ibm-team-repository-web-auth-msg “ and as long as it isn’t there, you’ve logged in.
- You need to save any cookies returned in the response and re-send those with your requests.
- To see if your login is still good (e.g. if you think your session timed out), hit /authenticated/identity on your server.
- If you’re logged in it will return a JSON object with your user name and roles. If not it will return the custom HTTP header with a value of “authrequired”.

About the API: Feeds

- The heart of the REST API is made up of a number of “feeds” (Atom/RSS feeds).
- Unlike a RSS feed that you’d subscribe to with a browser or Google Reader, these RSS feeds contain XML instead of HTML.
- Each feed is essentially a list of all instances of a certain type of artifact from RQM.

About the API: Feeds

- A typical RQM 4.0 server will have a large number of feeds to choose from:
  - –There is a master feed that lists all the projects on the RQM server.
  - –There is a master feed for each type of artifact (test case, test script, test plan, etc.) on the server. These feeds span across all projects on the server (new in 4.0).
  - –Each project has its own collection of artifact feeds (test case, test script, test plan, etc.).

About the API: Feeds

- The “global” artifact feeds are new to RQM 4.0, so if you work with RQM 3.0 or older you have to do everything on a per-project basis.
- The artifacts shown in a feed are not full copies of the actual artifact. Only important fields (e.g. description, name, ID) are included. Each entry contains a link you can follow to get the full artifact as XML.
- Feeds return between 50 and 500 items at a time. In order to get the next batch of items you have to find and follow the “next” link that is embedded at the top of the feed.
- You can change the number of items returned via a server-side setting. At LPS, I had my admins increase the feed size from 50 to 500. I found that while I had fewer downloads to do to cover a whole project each one took 10 times as long.
- Technically you can request all this as JSON also, but I have never tried that.

About the API: Feeds

- The .NET runtime has a nice class for quick parsing of feeds: System.ServiceModel.Syndication.SyndicationFeed
- http://msdn.microsoft.com/en-us/library/system.servicemodel.syndication.syndicationfeed.aspx
- Sadly, Java doesn’t have anything like that built in. I just use JAXB when I have to code in Java.

About the API: Feeds

- Project-specific Feed URLs are of the format
  - https://<server>:<port>/<context>/service/com.ibm.rqm.integration.service.IIntegrationService/resources/<project API name>/<resource type>
  - (e.g. https://rqm.example.com:9443/qm/service/com.ibm.rqm.integration.service.IIntegrationService/resources/SampleProject/testcase)
- The project API name is a URL encoded form of the project's *ORIGINAL* name.
  THERE IS NO WAY TO CHANGE THIS NAME.
- RQM 4.0's global feeds are similar. Just remove the <project API name> part.

About the API: Object Model

- See https://jazz.net/wiki/bin/view/Main/RqmApi#Detailed_Schema_Documentation for links to version-specific XML schema documentation that IBM generates and posts.
- (RQM 3.0: http://jazz.net/projects/rational-quality-manager/api-doc-3.0.1.6)
- (RQM 4.0: https://jazz.net/products/rational-quality-manager/api-doc-4.0/)

- You have to log in before you can download schemas from your server.

- The core XML Schema Document (XSD) is located on your RQM server at the relative path:
  - service/com.ibm.rqm.integration.service.IIntegrationService/schema/qm.xsd
  - If you log into https://rqm.example.com:9443/qm, the qm.xsd will be at https://rqm.example.com:9443/qm/service/com.ibm.rqm.integration.service.IIntegrationService/schema/qm.xsd

About the API: Object Model

- Since the login is forms auth based you cannot point tools like xjc.exe (Java) or xsd.exe (C#) at the URL and generate classes.
  1. You have to download qm.xsd manually and open it in a text editor.
  2. Near the top of the file there will be a bunch of <**xs:import namespace**="" **schemaLocation**=""/> tags.
  3. You have to download each one of the URLs shown in the **schemaLocation** attrributes by hand to the same directory as qm.xsd *and* edit the **schemaLocation** attribute to show just a file name.
  4. Once you've done all that then you can run xjc.exe or xsd.exe on the locally stored XSD files.
- I wrote a Perl script to do this for me. It's part of the sample code I've posted for download.

About the API: Object Model

- ***The contents of qm.xsd are not going to be exactly the same across user logins and/or RQM servers.***
- Sections will simply not be listed in the XSD if the user you logged in with doesn't have permissions to see those parts of RQM.
- If you add custom sections to a RQM project they will appear in the XSD with ***TAG NAMES*** like this: *com.ibm.rqm.planning.editor.section.dynamicSection_1342721266030*
- If you go into the RQM UI and manually add sections that are exactly alike to two projects, they will still have different XML tag names in the REST API.

About the API: Object Model

- The only workaround I could find was to set up the artifacts in one project and then use the REST API to copy the templates to the other projects that needed to be the same.
- ***This means that classes generated against qm.xsd on one RQM server are not going to be 100% compatible with a different server, even if the version is the same.***
- You have to be prepared to delve into raw XML if necessary.
- If your work will never deal with custom sections in a RQM artifact then this quirk will never impact you.

About the API: Object Model – feed.xsd

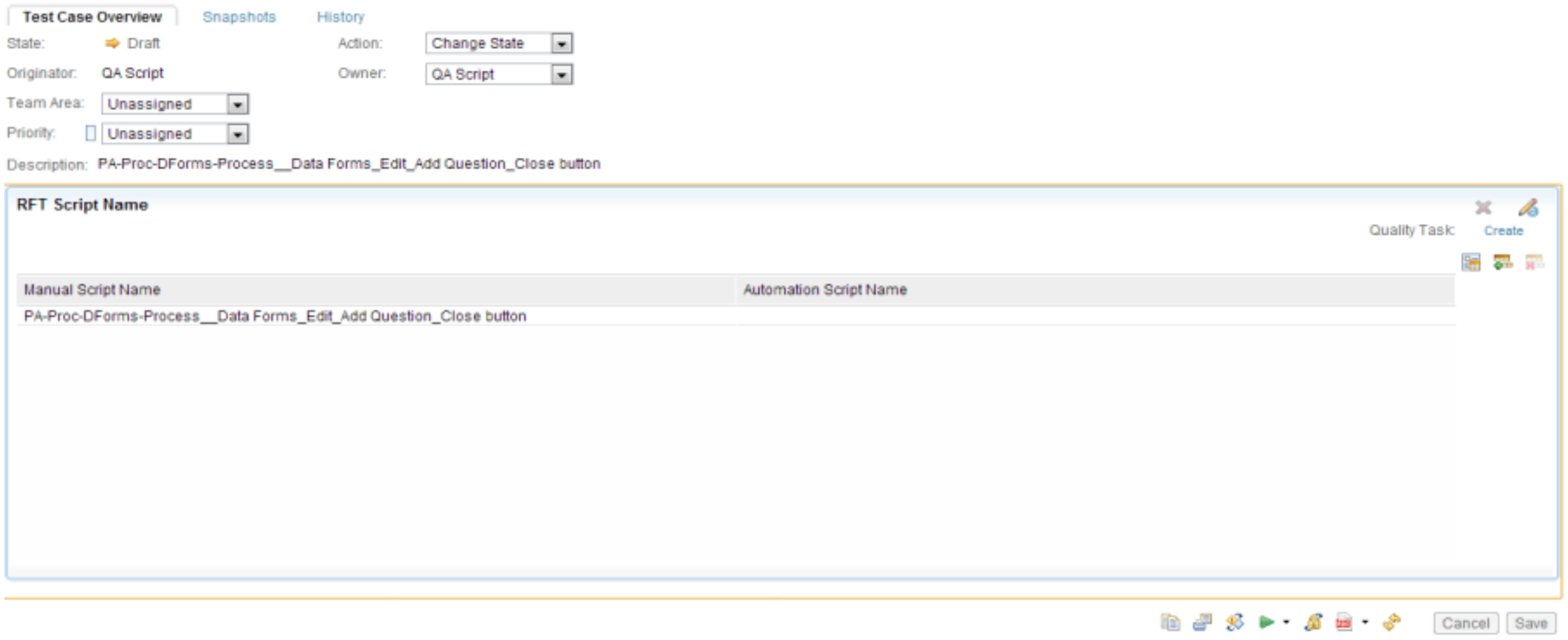
- Feed.xsd is the XML Schema Document for the RQM Atom feeds.
- If you use C# you can ignore it and just use the SyndicationFeed class I mentioned earlier.
- If you use Java/JAXB you have to modify it before compiling it into objects.
  - Delete this line: “**<xs:element maxOccurs="1" minOccurs="0" ref="qm:archived"/>**"

    For some unknown reason IBM didn't seem to define that tag in my copy of qm.xsd. Maybe after Innovate I will have time to open a PMR
  - Change "**<xs:element ref="xhtml:div"/>**" to "**<xs:any maxOccurs="unbounded" minOccurs="0" namespace="http://www.w3.org/1999/xhtml" processContents="skip"/>**"

    This will keep JAXB from wanting an XSD for XHTML, which apparently it can't parse out of the box anyway.

About the API: Object Model

- Some RQM custom section types (e.g. GRID) get rendered as JSON inside a single XML tag. See example below. It's a grid in the UI and JSON in a single XML tag.
- As far as I can tell, there is no "Content-Type" header you can supply to avoid this.



State:	Draft	Action:	Change State
Originator:	QA Script	Owner:	QA Script
Team Area:	Unassigned		
Priority:	Unassigned		

Description: PA-Proc-DForms-Process__Data Forms_Edit_Add Question_Close button

RFT Script Name

Manual Script Name	Automation Script Name
PA-Proc-DForms-Process__Data Forms_Edit_Add Question_Close button	

About the API: Object Model: Custom Sections JSON

```
<com.ibm.rqm.planning.editor.section.dynamicGridSection_1311015652479 xmlns="http://jazz.net/
xmlns/alm/qm/v0.1/"
extensionDisplayName="RFT Script Name"
type="GRID">
{ "columnNames":[ "Manual Script Name", "Automation Script Name" ],
   "value":[
      [
         ["PA-Proc-DForms-Process__Data Forms_Edit_Add Question_Close button"],
         [""]
      ]
   ]
}
</com.ibm.rqm.planning.editor.section.dynamicGridSection_1311015652479
xmlns="http://jazz.net/xmlns/alm/qm/v0.1/"
extensionDisplayName="RFT Script Name" type="GRID">
```

C# & the API

- I recommend using .NET 4.0 or newer if you can.
- Although it may look tempting, I do not recommend using System.Net.WebClient. You will have to use raw HttpWebRequest objects if you want to achieve support for multiple threads.
- The .NET runtime has a nice class for quick parsing of feeds: System.ServiceModel.Syndication.SyndicationFeed
  - http://msdn.microsoft.com/en-us/library/system.servicemodel.syndication.syndicationfeed.aspx

C# & the API

- XSD.exe will be your friend for parsing the RQM XML schemas and converting them into C# objects.
  - http://msdn.microsoft.com/en-us/library/x6c1kb0s%28v=vs.100%29.aspx
- Recommended command line (for RQM 4, once you download the .xsd files):

-
  ```
  xsd.exe /l:CS /f /c processinfo.xsd dcterms.xsd jazzprocess.xsd rdf.xsd
  tasks.xsd adapter.xsd vega.xsd testsuitelog.xsd testscript.xsd
  executionworkitem.xsd executionresult.xsd catalog.xsd dc.xsd alm.xsd qm.xsd
  ```
- Many generated classes will be essentially identical because the schemas make use of anonymous types a lot.
- In most cases, you could consolidate identical looking classes into one class, but that means you have to hand edit the code.

C# & the API

- When dealing with JSON in C#, DataContractJsonSerializer can be useful.
  - http://msdn.microsoft.com/en-us/library/system.runtime.serialization.json.datacontractjsonserializer(v=vs.100).aspx

- You have to build classes and mark their members with DataMemberAttribute in order for the serializer to parse the JSON into a C# object.

Java & the API

- JDK 6 and newer has a tool named xjc.exe – it's part of JAXB (so for JDK 5 or older you need to download JAXB)
  - http://docs.oracle.com/javase/6/docs/technotes/tools/share/xjc.html
- Once you download and scrub the .xsd files from your RQM server, all you have to do is run two commands:
  - `xjc.exe feed.xsd`
  - `xjc.exe qm.xsd`
- xjc.exe will open the other schemas as needed as long as they're in the same directory.
- It will generate classes in com.ibm.rqm.xml.bind automatically for you.

Java & the API

- For JSON I recommend Jackson
  - http://wiki.fasterxml.com/JacksonHome
- Just like with C# though you have to handle generating and sending HTTP requests yourself.
- I used Apache HTTPClient 4.1 for my samples.
  - http://hc.apache.org/httpcomponents-client-ga/

Poster addon for Firefox

- https://addons.mozilla.org/en-us/firefox/addon/poster/
- Best tool for basic experimenting with RQM – it allows you to customize a HTTP request from within Firefox.
- One catch with Firefox in general: If I bring up a feed from RQM in Firefox's feed viewer and click on a link to a specific artifact that artifact will load fine, but if I copy the link and then paste it into a new tab, RQM tries to redirect me to the UI.
  - Only way around this is to use User Agent Switcher to change my Firefox user agent to something like Googlebot or a made up string.

Poster addon for Firefox

- At LPS, I use it a lot for prototyping. My workflow is typically like this:
  - –1. Download artifact via Firefox browsing of feeds.
  - –2. Edit artifact in RQM UI.
  - –3. Download artifact again and note what's changed.
  - –4. Write code that produces appropriate XML and dumps it to a file.
  - –5. Use Poster to submit the request. If the request fails, tweak it in Poster till it works, and only THEN go find the bug in my code.
  - –6. Change my code to be able to directly post to RQM.

Implementation Demonstrations

- Firefox Poster add on
    - Find artifact in UI
    - GET and inspect it as XML
    - Update an attribute of the XML
    - PUT the changed XML back and see the change in the UI
- Java:
  - Hello World, RQM edition (shows user roles and available projects on server)
  - Get Artifact By Web UI ID: Shows how to create a Swing GUI and log into RQM and download XML
  - Create TCER and upload attachment: How to include results from an external tool in RQM like we do
- C#:
  - Time permtting, C# equivalents to all of those
- Python
  - Time permitting, at least a hello-world (login, show roles, show projects) program

Code Overview

- Java
  - –Eclipse Workspace
  - –2 Projects: RQM_Common and RQMExamples
  - –"RQM_Common" is called on by all the examples in "RQMExamples". Hopefully you can reuse it too.
- C#
  - –Visual Studio 2010 Solution (requires .NET 4.0)
  - –Each example is its own project.
  - –"RQM.API" is the core of my RQM API work and shared by all examples. Hopefully you can reuse it too.
- I tried to make sure the Java and C# codebases were as identical as possible

Code Overview

- Perl
  - One script: the XSD download/scrubbing script
  - You have to set 3 variables (user, password, server) in the top of the script to use it.

Demo

Summary and takeaways

- The REST interface is a fantastic way to automate and streamline common CLM activities
- Start Small!
- Getting comfortable with the objects, operations and quirks takes time
  - –Importing testcases was easy because all it required was modifying one type of artifact.
  - –Adding new TCERs took weeks of experimenting to get everything right.

Where to go to learn more

- CLM REST:
  https://jazz.net/wiki/bin/view/Main/RqmApi#Detailed_Schema_Documentation
- OSLC:
  http://open-services.net/
- Today's sample code will be mirrored (along with this PowerPoint) on www.advancedrft.com
  - That's my personal mirror of this talk (and past talks too).

Break

Innovate2013
The IBM Technical Summit

Questions

Innovate2013
The IBM Technical Summit

Daily Apple TV giveaway

- Complete your session surveys online each day at a conference kiosk or on your Innovate 2013 Portal!
- Each day that you complete all of that day's session surveys, your name will be entered to win the daily Apple TV!
- On Wednesday be sure to complete your full conference evaluation to receive your free conference t-shirt!

sponsored by
AllianceTech

Innovate2013
The IBM Technical Summit

Acknowledgements and disclaimers

Thank You

Innovate2013
The IBM Technical Summit