Sample code snippet.

/LinkLabelExport/src/com/ibm/rtc/sample/ChangeSetLabelExport.java

```
/*******************************************************************************
 * Licensed Materials - Property of IBM
 * (c) Copyright IBM Corporation 2010, 2015. All Rights Reserved.
 *
 * Note to U.S. Government Users Restricted Rights:  Use,
 * duplication or disclosure restricted by GSA ADP Schedule
 * Contract with IBM Corp.
 *******************************************************************************/
package com.ibm.rtc.sample;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;
import java.util.StringTokenizer;

import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

import net.jazz.oslc.cm.datamodel.ChangeRequest;
import net.jazz.oslc.utils.HttpUtils;
import net.jazz.oslc.utils.NamespaceContextMap;

import org.apache.http.HttpEntity;
```

```
import org.apache.http.HttpResponse;
import org.apache.http.auth.InvalidCredentialsException;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.HttpResponseException;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.entity.ContentProducer;
import org.apache.http.entity.EntityTemplate;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.protocol.HTTP;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.xml.sax.InputSource;

import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.Selector;
import com.hp.hpl.jena.rdf.model.SimpleSelector;
import com.hp.hpl.jena.rdf.model.Statement;
import com.hp.hpl.jena.rdf.model.StmtIterator;

/**
 * This example extracts the label of a Change Sets of Git Repo
 * This code is based on Example04 in OSLC Workshop at https://jazz.net/library/article/635
 *
 * A Jazz Team server must be started.
 */
public class ChangeSetLabelExport {
         static String server;
         static String login;
         static String password;
         HttpClient httpclient;
```

```
	static String projectAreaName;
	static List<String> wiID = new ArrayList<String>();
	static String gitserver;
	static String linkType;
	static String outputFile;

	public static void main(String[] args) {
		Properties props = new Properties();
		try {

	props.load(ChangeSetLabelExport.class.getResourceAsStream("params.properties"))
;
			ChangeSetLabelExport.server = props.getProperty("server.url");
			ChangeSetLabelExport.login = props.getProperty("login.id");
			ChangeSetLabelExport.password                                    =
props.getProperty("login.password");
			ChangeSetLabelExport.projectAreaName                             =
props.getProperty("pa.name");
			String ids = props.getProperty("cr.ids");
			StringTokenizer tk = new StringTokenizer(ids,  ",");
			while(tk.hasMoreTokens()) {
				ChangeSetLabelExport.wiID.add(tk.nextToken());
			}
			ChangeSetLabelExport.gitserver                                   =
props.getProperty("gitserver.path");
			ChangeSetLabelExport.linkType = props.getProperty("link.type");
			ChangeSetLabelExport.outputFile                                  =
props.getProperty("output.rdf");
		} catch (IOException e) {
			e.printStackTrace();
		}

		System.out.println(">> Extract ChangeSet Label as RDF");
		System.out.println("      - Server: "+ChangeSetLabelExport.server);
		System.out.println("      - Login: "+ChangeSetLabelExport.login);
		System.out.println("      - Password: *******");
```

```
                new ChangeSetLabelExport().run();
        }
        void run() {
                try {
                        // Step (1) : Retrieve the Service Providers catalog
                        String catalogURI = getServiceProviderCatalog();
                        System.out.println(">> Service Providers Catalog: "+catalogURI);

                        // Step (2) : Retrieve the designated Service Provider (Project
Area)
                        String   projectAreaURI   =   getServiceProvider(catalogURI,
ChangeSetLabelExport.projectAreaName);
                        System.out.println(">>              Project              Area
["+ChangeSetLabelExport.projectAreaName+"]: "+projectAreaURI);

                        // Step (3) : Retrieve the Change Request Simple Query for the
current Service Provider
                        String simpleQueryURI = getSimpleQueryURI(projectAreaURI);
                        System.out.println(">> Simple Query URL: "+simpleQueryURI);

                        // Step New : Extract Change Sets Label(title) property
                        // You can use dcterms:modified property to filter the data :
Example ?oslc.where=dcterms:modified>="2008-12-02T18:42:30"
                        Model outputModel = ModelFactory.createDefaultModel();
                        for (String id : ChangeSetLabelExport.wiID){
                                String queryURI = simpleQueryURI

        +"?oslc.where="+URLEncoder.encode("dcterms:identifier=¥""+id+"¥"", HTTP.UTF_8)
                                        +"&oslc.select=*";
//
        +"&oslc.select="+"dcterms:title,dcterms:identifier,rtc_cm:type,dcterms:descript
ion,dcterms:subject,dcterms:creator,dcterms:modified";
                                getChangeRequestAsRDF(queryURI, outputModel);
                        }
                        System.out.println(">> Extracted change sets.");
```

```
                        BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(new File(ChangeSetLabelExport.outputFile)));
                        outputModel.write(bos);

                        // Step (4) : Retrieve the designated Change Request (Work Item)
//                      ChangeRequest cr = getChangeRequest(simpleQueryURI, wiID);
//                      System.out.println(">> Change Request URL for ["+wiID+"]:
"+cr.getUri());

                        // Step (5) : Apply modification to the current Change Request
//                      cr.setDcDescription(cr.getDcDescription()+"      --      "+new
Date().toString());

                        // Step (6) : Update the Change Request on the server
//                      HttpResponse response = updateChangeRequest(cr);

                        // Step (7) : Print out the HTTP PUT method response
//                      System.out.println(">> Update Response Status code:" +
response.getStatusLine());
//                      System.out.println(">> Update Response Headers:");
//                      HttpUtils.printResponseHeaders(response);
//                      System.out.println(">> Update Response Body:");
//                      HttpUtils.printResponseBody(response);

                } catch (ClientProtocolException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                } catch (XPathExpressionException e) {
                        e.printStackTrace();
                } catch (InvalidCredentialsException e) {
                        e.printStackTrace();
                } finally {
                        // Shutdown the HTTP connection
                        httpclient.getConnectionManager().shutdown();
                }
```

```
        }
        ChangeSetLabelExport() {
                super();
                // Setup the HttClient
                this.httpclient = new DefaultHttpClient();
                HttpUtils.setupLazySSLSupport(httpclient);
        }
        /**
         * Upload the Change Request
         *
         * @param cr      - the Change Request to upload.
         */
        private HttpResponse updateChangeRequest(final ChangeRequest cr) throws
InvalidCredentialsException, IOException {
                // How to fill the request body (Content Producer)
                ContentProducer cp = new ContentProducer() {
                   public void writeTo(OutputStream outstream) throws IOException {
                       Writer writer = new OutputStreamWriter(outstream, HTTP.UTF_8);

                       cr.writeXML(writer);
                       writer.flush();
                   }
                };

                HttpEntity entity = new EntityTemplate(cp);
                HttpPut put = new HttpPut(cr.getUri());
                put.addHeader("Accept", "application/rdf+xml");
                put.addHeader("Content-type", "application/xml");
                put.setEntity(entity);

                // Call the PUT method against the Change Request URI
                return HttpUtils.sendPutForSecureDocument(

        ChangeSetLabelExport.server,      cp,      put,      ChangeSetLabelExport.login,
ChangeSetLabelExport.password, httpclient);
        }
```

```
        private void getChangeRequestAsRDF(String simpleQueryURI, Model outputModel) throws
InvalidCredentialsException, IOException, XPathExpressionException {
                // Build the query requesting a change request with a specific dc:identifier
                // Fetch only a subset of its properties
                String queryWIs = simpleQueryURI;

                HttpGet query = new HttpGet(queryWIs);
                query.addHeader("Accept", "application/rdf+xml");
                query.addHeader("OSLC-Core-Version", "2.0");

                HttpResponse response = HttpUtils.sendGetForSecureDocument(

                ChangeSetLabelExport.server,     query,     ChangeSetLabelExport.login,
ChangeSetLabelExport.password, httpclient);
                if (response.getStatusLine().getStatusCode() != 200) {
                        response.getEntity().consumeContent();
                        throw                                                    new
HttpResponseException(response.getStatusLine().getStatusCode(),
response.getStatusLine().getReasonPhrase());
                }
                // Create RDF model
                Model model = ModelFactory.createDefaultModel();
                model.read(new
InputStreamReader(response.getEntity().getContent()),null);
                model.write(System.out);

                // Select stmt
                Property                        predicateProp                        =
model.createProperty("http://www.w3.org/1999/02/22-rdf-syntax-ns#", "predicate");
                Resource                      tracksChangeSetObj                     =
model.createResource(ChangeSetLabelExport.linkType);
                Selector   selector   =   new   SimpleSelector(null,   predicateProp,
tracksChangeSetObj) {
                        public boolean selects(Statement stmt) {
                                Resource subject = stmt.getSubject();
```

```
                                Property                objectProp                =
stmt.getModel().createProperty("http://www.w3.org/1999/02/22-rdf-syntax-ns#", "object");
                                Statement stmt2 = subject.getProperty(objectProp);
                                RDFNode object = stmt2.getObject();
                                if (object instanceof Resource) {
                                        Resource res = (Resource)object;
                                        if
(res.getURI().startsWith(ChangeSetLabelExport.server+ChangeSetLabelExport.gitserver)) {
                                                return true;
                                        }
                                }
                                return false;
                        }
                };
                Property                        subjectProp                        =
model.createProperty("http://www.w3.org/1999/02/22-rdf-syntax-ns#", "subject");
                Property                        objectProp                        =
model.createProperty("http://www.w3.org/1999/02/22-rdf-syntax-ns#", "object");
                Property                        typeProp                        =
model.createProperty("http://www.w3.org/1999/02/22-rdf-syntax-ns#", "type");
                Property titleProp = model.createProperty("http://purl.org/dc/terms/",
"title");
                StmtIterator itr = model.listStatements(selector);
                List<Statement> statements = new ArrayList<Statement>();
                while(itr.hasNext()) {
                        Statement stmt = itr.nextStatement();
                        Resource sub = stmt.getSubject();
                        statements.add(sub.getProperty(subjectProp));
                        statements.add(sub.getProperty(predicateProp));
                        statements.add(sub.getProperty(objectProp));
                        statements.add(sub.getProperty(typeProp));
                        statements.add(sub.getProperty(titleProp));
                }
                outputModel.add(statements.toArray(new Statement[0]));
        }
```

```
        /**
         * Fetch a specific Change Request and map it to an instance of ChangeRequest
         *
         * @param simpleQueryURI   - query
         * @param wiID                                - Change Request ID
         */
        private ChangeRequest getChangeRequest(String simpleQueryURI, String wiID) throws
InvalidCredentialsException, IOException, XPathExpressionException {
                // Build the query requesting a change request with a specific dc:identifier
                // Fetch only a subset of its properties
                String queryWIs = simpleQueryURI

        +"?oslc.where="+URLEncoder.encode("dcterms:identifier=¥""+wiID+"¥"", HTTP.UTF_8)

        +"&oslc.select="+"dcterms:title,dcterms:identifier,rtc_cm:type,dcterms:descript
ion,dcterms:subject,dcterms:creator,dcterms:modified";

                HttpGet query = new HttpGet(queryWIs);
                query.addHeader("Accept", "application/xml");
                query.addHeader("OSLC-Core-Version", "2.0");

                HttpResponse response = HttpUtils.sendGetForSecureDocument(

                ChangeSetLabelExport.server,     query,     ChangeSetLabelExport.login,
ChangeSetLabelExport.password, httpclient);
                if (response.getStatusLine().getStatusCode() != 200) {
                        response.getEntity().consumeContent();
                        throw                                                    new
HttpResponseException(response.getStatusLine().getStatusCode(),
response.getStatusLine().getReasonPhrase());
                }

                // Define the XPath evaluation environment
                XPathFactory factory = XPathFactory.newInstance();
                XPath xpath = factory.newXPath();
```

```
                String wiXPath = "//oslc_cm:ChangeRequest";
                xpath.setNamespaceContext(
                                new NamespaceContextMap(new String[]
                                                {"oslc_cm",
"http://open-services.net/ns/cm#"}));

                // Parse the response body to retrieve the Change Request DOM node
                InputSource source = new InputSource(response.getEntity().getContent());

                Element    wiNode    =    (Element)(xpath.evaluate(wiXPath,    source,
XPathConstants.NODE));
                if (wiNode == null) {
                        response.getEntity().consumeContent();
                        throw new RuntimeException("Missing Change Request: "+wiID);
                }
                // Create the corresponding ChangeRequest instance
                String wiURI = wiNode.getAttribute("rdf:about");
                return new ChangeRequest(wiURI, wiNode);
        }
        /**
         * Return the SimpleQuery URI used to retrieve Change Requests
         *
         * @param projectAreaURI  - Service Provider owning the Change Request
         */
        private     String     getSimpleQueryURI(String     projectAreaURI)     throws
InvalidCredentialsException, IOException, XPathExpressionException {
                HttpGet query = new HttpGet(projectAreaURI);
                query.addHeader("Accept", "application/xml");
                query.addHeader("OSLC-Core-Version", "2.0");

                // Access to the Service Descriptor document
                HttpResponse response = HttpUtils.sendGetForSecureDocument(

                ChangeSetLabelExport.server,     query,     ChangeSetLabelExport.login,
ChangeSetLabelExport.password, httpclient);
                if (response.getStatusLine().getStatusCode() != 200) {
```

```
                        response.getEntity().consumeContent();
                        throw                                                    new
HttpResponseException(response.getStatusLine().getStatusCode(),
response.getStatusLine().getReasonPhrase());
                }
                // Define the XPath evaluation environment
                XPathFactory factory = XPathFactory.newInstance();
                XPath xpath = factory.newXPath();
                String                          simpleQueryXPath                          =
"//oslc:QueryCapability/oslc:queryBase/@rdf:resource";
                xpath.setNamespaceContext(
                                new NamespaceContextMap(new String[]
                                                {"oslc",
"http://open-services.net/ns/core#",
                                                "rdf",
"http://www.w3.org/1999/02/22-rdf-syntax-ns#"}));

                // Parse the response body to retrieve the Change Management Simple Query
URL
                InputSource source = new InputSource(response.getEntity().getContent());
                Node simpleQueryURLNode = (Node)(xpath.evaluate(simpleQueryXPath, source,
XPathConstants.NODE));
                if (simpleQueryURLNode == null) {
                        response.getEntity().consumeContent();
                        throw new RuntimeException("Missing simpleQuery element!");
                }
                return simpleQueryURLNode.getTextContent();
        }
        /**
         * Retrieve the URI of a specific Service Provider
         *
         * @param catalogURI       - The catalog to look into
         * @param paName           - The Service Provider we are looking for
         */
        private  String  getServiceProvider(String  catalogURI,  String  paName)  throws
InvalidCredentialsException, IOException, XPathExpressionException {
```

```
                HttpGet query = new HttpGet(catalogURI);
                query.addHeader("Accept", "application/xml");
                query.addHeader("OSLC-Core-Version", "2.0");

                // Access to the Service Providers catalog
                HttpResponse response = HttpUtils.sendGetForSecureDocument(

                ChangeSetLabelExport.server,     query,     ChangeSetLabelExport.login,
ChangeSetLabelExport.password, httpclient);
                if (response.getStatusLine().getStatusCode() != 200) {
                        response.getEntity().consumeContent();
                        throw                                                    new
HttpResponseException(response.getStatusLine().getStatusCode(),
response.getStatusLine().getReasonPhrase());
                }
                // Define the XPath evaluation environment
                XPathFactory factory = XPathFactory.newInstance();
                XPath xpath = factory.newXPath();
                String                        serviceProviderXPath                        =
"//oslc:ServiceProvider[dcterms:title=¥""+paName+"¥"]/@rdf:about";
                xpath.setNamespaceContext(
                                new NamespaceContextMap(new String[]
                                                {"oslc",
"http://open-services.net/ns/core#",
                                                "rdf",
"http://www.w3.org/1999/02/22-rdf-syntax-ns#",
                                                "dcterms",
"http://purl.org/dc/terms/"}));

                // Retrieve the designated Service Provider
                InputSource source = new InputSource(response.getEntity().getContent());
                Node   paNode   =   (Node)(xpath.evaluate(serviceProviderXPath,   source,
XPathConstants.NODE));
                if (paNode == null) {
                        response.getEntity().consumeContent();
                        throw new RuntimeException("Unknown Project Area: "+paName);
```

```
                }
                return paNode.getTextContent();
        }
        /**
         * Retrieve the Service Provider Catalog URI from the Root Services Document
         */
        private String getServiceProviderCatalog() throws InvalidCredentialsException,
IOException, XPathExpressionException {
                String rootServices = ChangeSetLabelExport.server + "/rootservices";
                HttpGet query = new HttpGet(rootServices);
                query.addHeader("Accept", "application/xml");
                query.addHeader("OSLC-Core-Version", "2.0");

                // Request the Root Services document
                HttpResponse response = HttpUtils.sendGetForSecureDocument(

                ChangeSetLabelExport.server,     query,     ChangeSetLabelExport.login,
ChangeSetLabelExport.password, httpclient);
                if (response.getStatusLine().getStatusCode() != 200) {
                        response.getEntity().consumeContent();
                        throw                                                    new
HttpResponseException(response.getStatusLine().getStatusCode(),
response.getStatusLine().getReasonPhrase());
                }
                // Define the XPath evaluation environment
                XPathFactory factory = XPathFactory.newInstance();
                XPath xpath = factory.newXPath();
                String                          catalogXPath                          =
"/rdf:Description/oslc_cm:cmServiceProviders/@rdf:resource";
                xpath.setNamespaceContext(
                                new NamespaceContextMap(new String[]
                                                {       "rdf",
"http://www.w3.org/1999/02/22-rdf-syntax-ns#",

        "oslc_cm","http://open-services.net/xmlns/cm/1.0/"}));
```

```
                // Parse the response body to retrieve the catalog URI
                InputSource source = new InputSource(response.getEntity().getContent());
                Node attribute = (Node) (xpath.evaluate(catalogXPath, source,
XPathConstants.NODE));
                String serviceProvidersCatalog = attribute.getTextContent();
                response.getEntity().consumeContent();

                return serviceProvidersCatalog;
        }
}
```

- /LinkLabelExport/src/com/ibm/rtc/sample/params.properties

```
server.url=https://rtcserver.mycompany.com:9443/ccm/
login.id=susan@mycompany.com
login.password=mypassword
pa.name=ProjectArea
cr.ids=7,8,13
gitserver.path=com.ibm.team.git.internal.resources.IGitResourceRestService/commit
link.type=http://open-services.net/ns/cm#tracksChangeSet
output.rdf=c:/tmp/changesets.rdf
```

- /LinkLabelExport/.class

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
        <classpathentry kind="src" path="src"/>
        <classpathentry kind="con"
path="org.eclipse.jdt.launching.JRE_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher
.StandardVMType/JavaSE-1.6"/>
        <classpathentry combineaccessrules="false" kind="src"
path="/net.jazz.oslc.consumer.cm.client"/>
        <classpathentry combineaccessrules="false" kind="src" path="/org.apache.http"/>
        <classpathentry combineaccessrules="false" kind="src" path="/org.apache.jena"/>
```

```
        <classpathentry kind="output" path="bin"/>
</classpath>
```

- /org.apache.jena/.class

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
        <classpathentry                                                    kind="con"
path="org.eclipse.jdt.launching.JRE_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher
.StandardVMType/JavaSE-1.6"/>
        <classpathentry exported="true" kind="lib" path="jena-arq-2.9.1.jar"/>
        <classpathentry exported="true" kind="lib" path="jena-core-2.7.1.jar"/>
        <classpathentry exported="true" kind="lib" path="jena-iri-0.9.1.jar"/>
        <classpathentry exported="true" kind="lib" path="jena-tdb-0.9.1.jar"/>
        <classpathentry exported="true" kind="lib" path="slf4j-api-1.6.4.jar"/>
        <classpathentry exported="true" kind="lib" path="slf4j-log4j12-1.6.4.jar"/>
        <classpathentry exported="true" kind="lib" path="log4j-1.2.16.jar"/>
        <classpathentry kind="output" path="bin"/>
</classpath>
```