Status Report for ESG-SRM Project

The objective of the project is to design a method enabling the ESG2 project to access the data stored on HPSS using the Berkley Storage Manager (BeStMan). For this we utilized the BeStMan api that came with BeStMan-2.

So far we have implemented SRM-Get and SRM-Ls functions. It was assumed that only these two functions would be needed for requests made through ESG front end.

**Instructions to set up the project:**

1. Obtain source from git repository:

git clone [git@github.com:jfharney/esg-srm](mailto:git@github.com:jfharney/esg-srm)

1. Change to esg-srm directory

cd esg-srm

1. For eclipse:

cp project.txt .project

In eclipse import project from the git project.

Properties>java build path>source : Change source to src/java/main

1. Create directory

mkdir etc

mkdir etc/etc

1. Make project

ant make\_war (sudo may be needed)

1. In eclipse, add external libraries to the build path:

All jars from external\_jars/BeStManClient folder

All jars from lib/fetched/compile folder

1. Deploy dist/esg-srm.war.

**TODO:**

Deployed war unable to use httpg protocol to connect to the srm server:

Connecting to serviceurl : httpg://esg2-sdnl1.ccs.ornl.gov:46790/srm/v2/server  
SRM-CLIENT: got remote srm object  
java.net.MalformedURLException: unknown protocol: httpg  
        at java.net.URL.<init>(URL.java:574)  
        at java.net.URL.<init>(URL.java:464)  
        at java.net.URL.<init>(URL.java:413)  
        at gov.lbl.srm.client.wsdl.SRMServer.connect(SRMServer.java:432)  
        at org.esgf.srm.SRMRequestObject.runBeStManGetRequest(SRMRequestObject.java:282)  
        at org.esgf.srm.SRMRequestController.getSRMRequest(SRMRequestController.java:33)  
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)  
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)  
        at java.lang.reflect.Method.invoke(Method.java:597)  
        at org.springframework.web.bind.annotation.support.HandlerMethodInvoker.invokeHandlerMethod(HandlerMethodInvoker.ja  
va:176)...

But this problem doesn’t occur when the program is run using the command line. Probably a limitation on the tomcat server end.

**Instructions for generating certificate manually:**

1. Remove existing user certificate:
   1. rm /tmp/x509up\_u<userId>
2. Generate new certificate (requires passphrase, need to be a registered user):
   1. /usr/local/esg2/globus/bin/myproxy-logon –s esg2-gw –l <username in esg gateway>

**Daemon: To be completed**

**Before running BeStMan queries, we must obtain credentials to be able to do so. These credentials expire every 12 hours and need to be renewed. We need to write a daemon process that would do so automatically.**

Program Flow:

A request made is passed to the SRMRequestController as an HttpServletRequest object from which we extract the following query parameters:

1. url: The URL of the requested file in the SRM storage.
2. email: The email id of the requester to which we send the confirmation of receipt and completion of the request.
3. openid: The Open Id of the user.
4. proxyid: Proxy of the user
5. pass: Password of the user for the proxy.

Among the above parameters, so far we have used only the first 2. Although the proxy and password will be needed to run BeStMan queries, we haven’t chalked out a way to do so yet. Currently we are running BeStMan queries as a user with credentials to access BeStMan.

**Note:**

**Proxy Id and proxy password are to be used to generate the user certificate. We may also create a separate authentication layer and make srm calls as a single user in the backend. Currently all srm calls are made as user “e1g” user id “8048”.**

First break the URL to extract the srm *server url* which is the portion of the url that comes before the ‘?’.

The entire url (including server url) forms the *source url* (surl).

Form the Get or Ls request using the server url and the surl.

**SRM Get:**

Following the method samples provided with the API, first establish a connection with the server and then submit the request.

Next, send an email notification to the user at the email id provided conforming that his/her request has been submitted. We use the java mail api for this.

Then periodically check the status of the request (polling) as long as we have confirmation of success or failure. After that send an email to the user notifying him/her accordingly.

After successful run of SRM-Get, we extract the *transfer url* (turl) from which we may download the file that has been copied to the SRM cache using any transfer protocol such as http.

Multiple URLs may be sent as parameter using ‘;’ to separate them. Must belong to the same SRM server.

**NOTE:**

**The return link is currently a gsiftp link. We need to convert it to a http link.**

**SRM Ls:**

I have implemented the SRMLs functionality corresponding to SRMDirTest.java file provided in the BeStMan API. We haven’t tied it up with user request yet due to certain uncertainties about the workflow. Will complete that soon.

**Java Mail API:**

To utilize the java mail api to send mail, we need the following:

1. An SMTP host: smtp.ornl.gov
2. SMTP port: 25
3. A user account on the host: need to get a permanent account.
4. User id: The user id of that account. When using personal ORNL mail account, user id is the UCAMS id.
5. Password
6. Transfer Protocol: Set to SSL.
7. To email id: The user’s email id to which email is to be sent.

Emails are sent by an object of the EmailNotifier class. The constructor for this class takes arguments in the form of an array of Strings as parameters with values:

-to <user’s email id> default: None

-from<the ornl email id>: default: Set to service’s email address. Currently my email address

-user <user id>: default: Set to service’s user id. Currently my UCAMS

-subject <subject of the email>: default: “Your Request has been submitted/completed”

-body <body of the email>: contains the URL. Body of the email is generated in SRMGetObject.

-host <host smtp server> : default: smtp.ornl.gov

-port <smtp port usually 25> : default: 25

-password <password for the email account>: service’s user password

Call function sendMail().

NOTE:

We are unable to send mail from the sdnl1 machine currently. We need to ask ops to set up mail on the machines.

IMPORTANT:

Error on tomcat log files: Print statements and stack trace aren’t reflected on the tomcat log files. The last working commit in which the stack trace was seen in the log files was: commit 0a2414ea2955454e2f08e8c1cda4809afda99e20.

Posts on httpg error:

<http://mail-archives.apache.org/mod_mbox/tomcat-users/200903.mbox/%3c8baedd360903110412g743ddd43t263ecb513e08b004@mail.gmail.com%3e>

http://osdir.com/ml/users-tomcat.apache.org/2009-03/msg00617.html