GGF International Summer School on Grid Computing 2005 (ISSGC05)

Day 5 Exercise – Using established Grid Services

1 Introduction

This exercise will follow the GT4 tutorial in which you will have developed a range of GT4 skills. It should let you build on that experience and return to the progressive exercise. It is not necessary to complete every part of this exercise.

Here you apply what you have learned about the GT4 toolkit and WSRF by converting your Explorer client to use stateful services that offer similar functions to those used in Day 2. You can then test this on one of the surfaces modelled on Tuesday and optionally employ your Explorer on one or two new surfaces.

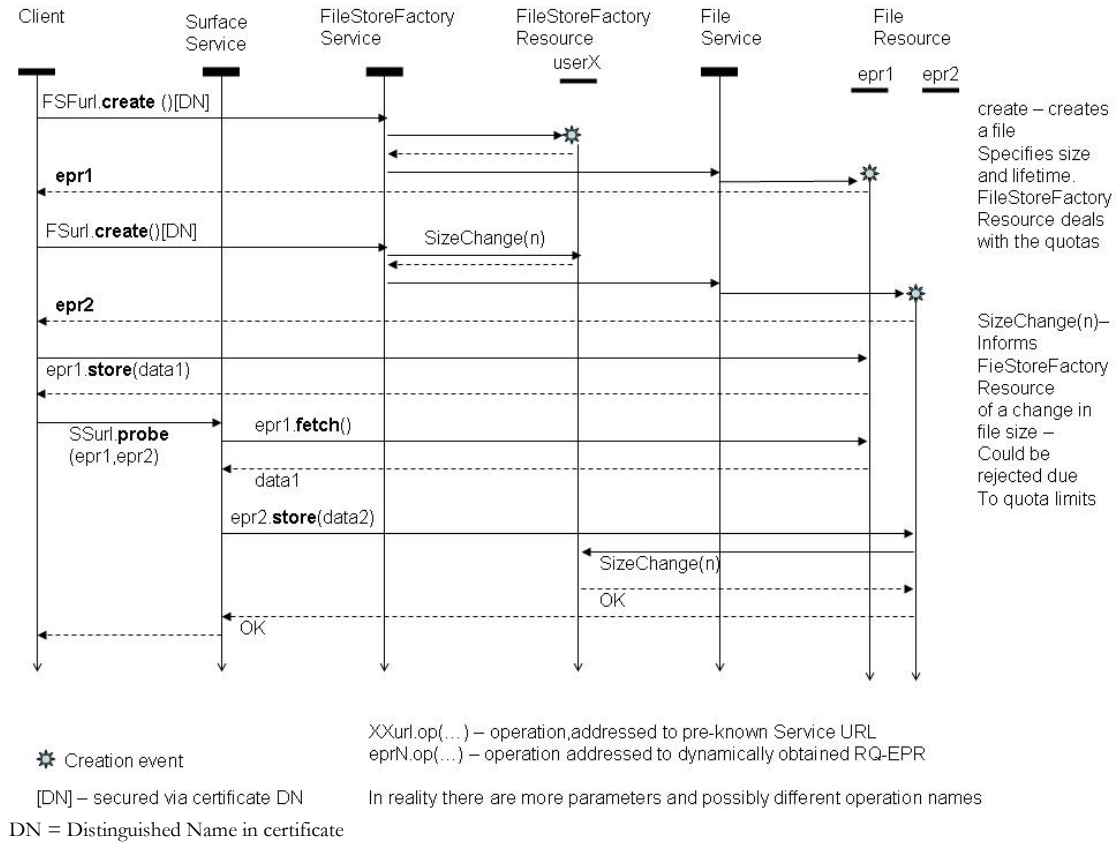
When you have completed this exercise, you should have

1. Experience of reading WSDL to understand the functionality of a service
2. Gained experience of using stateful Services which incorporate the WSRF mechanisms
3. Started to develop an initial appreciation of resource allocation issues in Grids

2 Background Information

Compared with the web services from Day 2, the stateful services used here exhibit significant changes in the structure and interfaces. Some particular changes are:

- Each user now has a file-space Allocation, as the maximum number of kilobytes of file space that they can be using at any one time. Each file has a lifetime and supports scheduled termination.
- The `FileStore` service has become two stateful services - `FileStoreFactory` and `FileService`. The diagram below shows how these services and their resources operate in one possible usage. This will be explained in a talk introducing the exercise
- Before storing data into a file it has to be created, as a separate operation. The create operation specifies a file size. This has the effect of reserving the space for the file and so reduces the available file space. However when the file is actually stored, the size may be different from that reserved, with the corresponding effect on the remaining available file space.



3 Stages in the Progressive Exercise

All of the work in this stage can be done as group work, and you might consider different team members working on different steps in parallel.

Step 3.1 Install provided software

We have provided a number of stateful services corresponding the previous web services, and a number of "test" clients which exercise those services. Here you will install the test clients and check that they work.

1. **mkdir /tmp/part2/**
2. **cd /tmp/part2**
3. **cd ~lcc/part2**
4. **cp exercise-install2.tar.gz /tmp/part2/exercise-install2.tar.gz**
5. **cp gt-install2 /tmp/part2/gt-install2.tar.gz**
6. **cd /tmp/part2**
7. **tar xzf gt-install2.tar.gz**
8. **cd gt-install2**
9. **export GLOBUS_LOCATION=$PWD**
10. **echo $GLOBUS_LOCATION**
    check that it is **/tmp/part2/gt-install**
11. **cd /tmp/part2**
12. **tar xzf exercise-install2.tar.gz**
13. **cd gridSchool0705**

14. from the gridSchool0705 directory, edit the file
    ws-clients/ws-clients/src/org/globus/tutorial/client/FileStoreClient.java
    making two changes:
    a. change **http://localhost:8080/wsrf/services/FileStoreFactoryService**
       to **http://192.167.2.4:8111/wsrf/services/FileStoreFactoryService**
    b. change **http://localhost:8080/wsrf/services/FileService**
       to **http://192.167.2.4:8111/wsrf/services/FileService**
15. from the gridSchool0705 directory, do **ant deploy**

Note you should not proceed unless this step 15 ends with "**build successful**" being output.

16. Generate a proxy by executing the following commands:
    a. **source $GLOBUS_LOCATION/etc/globus-devel-env.sh**
    b. **java org.globus.tools.ProxyInit**
17. **cd ws-clients/ws-clients**
18. **ant fileStoreClient**

Note you should not proceed unless this step 13 ends with "**build successful**" being output. The ant build also automatically runs the client that has been built. This test client outputs various messages pertaining to the clients it is testing. It runs a test on each opertation of each client.

Step 3.2 Port Your Explorer Client

Here you will make the minimum necessary changes to you Explorer program so that it now operates using the resourceful Services, build it and test it out using the `Surface1` Service.

The strategy we suggest for doing the port is

- Read the portytype definitions (and associated message types etc.) in the WSDLs for the provided services, in order to understand where the functionality has changed.
- Read the Java code for the provided test clients in order to understand how to invoke the new services.
- Use the code for your existing client and parts of the code for the test clients to construct a new explorer client which has the same functionality as the previous Explorer client, but now using the stateful services.

In detail the steps for building your new client are:

1. **cd /tmp/part2/gridSchool0705/ws-clients/wsclients/src/org/globus/tutorial/client**
   The above is all one line
2. create a new file in this directory called Explorer.java with your new Explorer client code
3. add the new client file as a build target in
   /tmp/part2/gridSchool0705/ws-clients/ws-clients/build.xml
4. Invoke the explorer client

Step 3.3 Developing the Explorer Client

Having obtained a least-effort port of your Explorer client, you now need to develop it to be more appropriate to its new environment. This is open-ended, your criteria should be:

- what you think may be useful when you start to use your client as a tool for substantial surface exploration work latter in the course.
- what provides you with experience in the learning domain of WSRF.

We suggest, e.g.

- allowing for the separation of file creation from data storage. The reason for this would be so that file space can be pre-allocated, since otherwise there is the danger that a job will fail at the end due to lack of available file space for its output file, thus wasting the time that was taken in the execution of the job. At this stage, this may not be a significant consideration as the computations have low execution times. However in later stages (and in "real life") execution times may be much longer.
- dealing with the additional exceptions that now can be thrown by the service stubs.
- extending your CLI to provide file management facilities, such as file deletion.

Building each new version of the Explorer client will of course require the steps 3.2.1 – 3.2.4 described above.

Step 3.4 Using your Client to obtain data

Use the client you have built to explore the surface provided by the service `cone`. Find the cone. Estimate its height and the coordinates of its centre.

You are not expected to write an algorithm to find these values, rather you should perform a coarse scan to locate the cone, and then direct successive scans using your client until you have a number of samples in a small area including the apex. By displaying this small sample you should be able to estimate the coordinates and height above the surrounding plane of the apex.

You can, of course, automate this iterative search if you wish.

Step 3.5 Extending your Client to obtain measurements

This is an optional extra for teams that are progressing rapidly. The code that you develop will be useful in an exercise tomorrow.

Use the client you have built to explore the surface provided by the service `cuboid`. Find the cuboid. Estimate its dimensions and the coordinates of its min_x, min_y (bottom right hand) corner using a strategy similar to that used in Step 3.4.

Now modify your client to automate the search and measurement, assuming that the cuboid is aligned with the axes. That is, it should do a (random?) scan to find locations where there are coordinates above the plane. Then perform progressively more fine-step traversals to locate the edges precisely. Return `x_min`, `y_min` and `height`, where: `height` is the height of the cuboid above the surrounding plane; `x_min` and `y_min` are the co-ordinates of the cuboids bottom left corner. If you have time, also determine the coordinates of the cuboids top right corner.