caGrid migration to final WS-\* standards

# General Considerations

caGrid as it is currently built relies on Globus 4.0.3 to provide the Web Services and Web Services Resource Framework (WSRF) platform and tooling. Globus 4.0.3, however, utilizes draft versions of the assorted WS-\* specifications. More recently developed web service frameworks utilize the final versions of these specifications, so a migration would facilitate greater interoperability.

caGrid makes heavy use of WSRF to provide stateful services and a user-oriented view of the web service. A change to this fundamental assumption would require a complete rearchitecture of caGrid, and is beyond the scope of this investigation. The Globus Toolkit is one of a few implementations of WSRF, and enjoys a good history of success. The latest 4.x release of the Globus Toolkit is 4.2.1, and that release fully supports and implements the final WS-\* specifications. While the Globus Toolkit version 4.2.1 is more or less backwards compatible at an API and functional level with version 4.0.3, it is important to note that it is NOT backwards compatible at a protocol level, and therefore not interoperable with existing services built on Globus 4.0.3, which includes all current caGrid services. GT 4.2.1 utilizes the final release versions of the various WS-\* and WSRF specifications, while 4.0.3 uses primarily draft versions. These are not namespace equivalent, and contain slightly different features and XML payloads.

The Globus wiki contains a fairly comprehensive upgrade guide which identifies the schema and WSDL changes, as well as any API differences: <http://dev.globus.org/wiki/Java_WS_Core/Final_WSRF_Migration_Guide>. The most important API difference is the internalization of Axis to Globus, and the associated change in package names.

# Schema / WSDL change specifics

caGrid services generate a WSDL for each service context the grid service exposes, and any imported WSDLs which provide pre-built operations. The simplest grid service will have two WSDLs: One for the primary service itself, and one for the Service Security operations.

These WSDLs need to be updated to reference the new namespaces described in the migration guide, and the new XSD and WSDL file locations. It seems that these changes can be automated for an upgrade process, and are simply part of a template for newly created Introduce generated grid services.

The ResourceUnknownFault and ResourceUnknownFaultType are now defined in their own namespace for the WS-Resource specification. They were previously defined in both lifetime and resource. WSDLs and XSDs that reference these faults need to be changed accordingly.

Namespace excludes have to be updated to reflect the new namespaces so that stubs aren’t built for them which will conflict with those provided by Globus. This will be in the build-stubs.xml, and the introduce.properties file.

The namespace2package.mappings need updated with the new namespaces as well.

The addressing schema is referenced in the <ServiceContext>\_registration.xml and needs to be updated.

# API change specifics

# MDS / Index Service

The new APIs for the Index Service are not part of the Globus 4.2.1 distribution out of the box. The source code for MDS can be obtained and built from a CVS checkout:

cvs.globus.org/home/globdev/CVS/globus-packages

Everything has a tag for “Globus\_4\_2\_1”, so that’s what I got

Checkout the ws-mds, wsrf, and authorization modules as siblings to one another.

Set the GLOBUS\_LOCATION environment variable to the location of the globus toolkit 4.2 installation.

Execute “ant deploy –Dversion=4.2.1” in ws-mds first.

Execute “ant deploy –Dversion=4.2.1” in wsrf next.

# Security API Changes

A number of the APIs for authorization and authentication have changed. New packages were created and code generally refactored.

<http://www.globus.org/toolkit/docs/4.2/4.2.1/security/wsaajava/mig/index.html>

# General-purpose ServiceSecurityClient

This class is the base class of Introduce generated grid services, and needs to be updated to use the new namespaces and APIs.

# Globus on the classpath

The Globus lib directory has some additional subdirectories, which aren’t usually picked up by the ant built scripts. Change the scripts that reference GLOBUS\_LOCATAION/lib/\*.jar to GLOBUS\_LOCATION/lib/\*\*/\*.jar.

# Globus 4.2.1 Distribution

The Globus 4.2.1 WS-Core zip file that is obtainable from the Globus web site seems to be missing several important packages which caGrid makes extensive use of, and contains libraries which reference classes that were present in the 4.0.3 version, but have since been removed from the 4.2 distribution.

Examples:

1. None of the MDS libraries are present in the distribution. These are required for interaction with the index service
2. Some of the authentication libraries still reference classes in the org.apache.axis… package. This was internalized to org.globus.axis.

It seems the best way to overcome these problems is to grab a checkout of the Globus packages source code from their CVS repository and build it locally. This means we’ll be creating our own distribution of Globus 4.2.1, much as we did for 4.0.3 to include WS-Enumeration and some bugfixes for CoG-JGlobus.

Very Rough LOE for caGrid Migration to final WS-\* Specs with Globus 4.2.1

1. Schema / WSDL changes
   1. Rework Introduce service templates and skeleton code
      1. 1 wk
   2. Create upgrader in Introduce to migrate a service from old to new schemas and WSDL
      1. 2 wk
   3. Upgrade and test all the core caGrid services
      1. 1 wk
   4. *Total: 4 wks*
2. API Changes *(note: I don’t fully have all of these sorted out yet, and more may come up, especially in the security APIs)*
   1. Finish scoping out changes and API mappings
      1. 1.5 wk
   2. Introduce service templates and skeleton code
      1. 1 wk
   3. Upgraders for Introduce services
      1. 2 wk
         1. *If it is determined this is possible at all. That process accounts for most of the 2 weeks*
   4. Upgrade and test all the core caGrid services
      1. 2 wks, with special attention to security
   5. *Total: 6.5 wks*
3. Globus distribution
   1. Find all the Globus code required to build a workable distribution
      1. 1 week
   2. Audit changes made to caGrid’s distribution of Globus 4.0.3 to verify the changes and fixes made it into 4.2. Bring them in if not.
      1. 2 weeks
   3. Build all the code into a workable package
      1. 1 week
   4. *Total: 4 weeks*

I come up with an optimistic total of 12.5 weeks.