OGCE Developers Meeting

September 9-10

Bloomington, IN

Meeting Summary and SC Plans

These slides written Friday during/after PI meeting. In case of content conflict with other sections, this section takes precedence.

Supercomputing Punchlist

- Fix API for data share (IU, UMich)
- CoG4 Into uPortal: No assemply. Has CoG4, uPortal, and HypersonicSQL ()
- 168 MyProxy (IU)
- 168 Gram (IU)
- 168 GridFTP (IU)
- GPIR (TACC)
- Jetspeed2 and GridSphere with the above (TACC, IU)
- uPortal grid login (TACC)
- OGCE uPortal skin (UMich, SDSU): use the NMI site as a template
- Velocity 168 Gateway (IU-UMich)

Supercomputing Plan

- KISS: demo core grid portlets in 2-3 containers. Sequence shows priority
  - These use CoG 4 and show GT4 capability.
  - Portlets: MyProxy, GRAM, GPIR, GridFTP
  - Containers: uPortal, GridSphere, Jetspeed2
- Point to other demos using our stuff as application portals.
  - Don’t confuse the releases. Keep the grid stuff clean
- Show working for different GTs (see table)
- Lower Priority:
  - VelocityPortlet Converter for JSR 168.
  - JSR168WSRP portlet.
  - One button grid portal install

Globus/ Portal	GT2	GT3.2.1 (“classic” services”) GridFTP, GRAM, MyProxy	GT4 (“Classic services” are the same as 3.2.1) Demo GRAM
uPortal	X	X	X
GridSphere		x	x
Jetspeed			x

Out of the Box Grid Portal

- Need a plan
- Need a really simple build that runs all of the Grid stuff out of the box along with the portal at the same time.
- Can all run on a desktop.
- Set up to trust your default CA.
- Steps
  - Issue yourself a cert with the CoG.
  - Host certificate
  - Create a map file
  -

Meeting Slides

These were used in Thursday presentations

Thursday Agenda

9-10	JSR 168 review and critique (Marlon)
10-10.5	Developing 168 Grid Portlets (Marcus)
10.5-11	Using new CoG interfaces (Gregor)
11-Lunch	Sakai update (Chuck).
12-1	Overview of potential solutions to problems (Marlon)
1-5	Concurrent sessions –Hands on with Sakai •Modifying to add MyProxy •Modifying to add Joe's metadata repository –Hands on with uPortal and Grid Portlets •Modify to use Gregor's new interface. •Deploy in Jetspeed2, Gridsphere.
4-5	Brainstorming session: how to handles services in 168 portlets; how to support different development models; how to grid enable WSRP.

Friday Agenda

9-12	Concurrent sessions: –Continue developing grid portlets for GridFTP, etc. –Review of SC04 plans –Outreach and project follow-on plans –Optional presentations: HttpUnit and JSF
12-1:30	Marching orders
2:00-4:00	Informatics colloquium at IMU (free). –Joe Futrelle, Joseph Hardin, and I are speaking.

Goals for the Meeting

- Agree on strategy and plan for SC2004.
  - JSR 168 Grid Portlets working in multiple containers.
    - Gridsphere, uPortal, Jetspeed2
  - Agree upon strategy for service problems, particularly credential management.
  - Agree upon strategy for supporting various developer frameworks.
- Determine long term plan for 6 months, 12 months.
  - CHEF/Sakai compatibility through WSRP?
  - How do you “Grid Enable” WSRP?
- Develop and deploy prototype Grid portlets while we are here.

Motivating Issues

- We need to find a way to simplify life for other developers.
  - Science portal application developers will need to develop their own portlet interfaces, will have some biases about how to do this.
- We have many specific portlet components that can be converted to run in JSR 168 containers.
  - I want to review how to do this as foundational material and spend some development time
  - I also see several opportunities for OGCE, since we are "portlet vendors" rather than "container vendors".
- Gregor has developed higher level interfaces to the CoG for use in the portal.
  - We need to use and advertise.
- The original plan was to re-synch with CHEF/Sakai through uPortal/JSR168.
  - Sakai plans have changed to focus instead on WSRP.
  - WSRP and JSR168 are not incompatible.
  - Doing this for Grid portals will require security enhancements.

A JSR 168 Tutorial

This provides some necessary background information for discussion.

What is JSR 168?

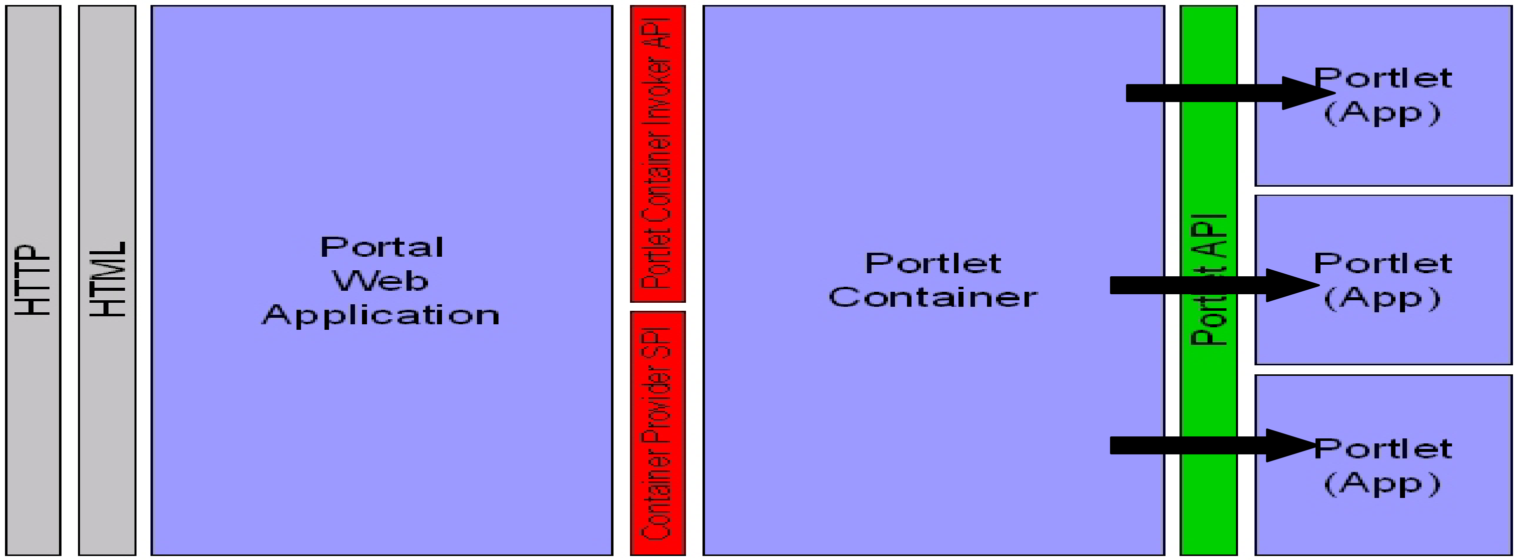
- (Indulge me if you know material in the next few slides--I need to make some arguable points).
- From the portlet development point of view, it is really very simple:
  - You write a java class that extends GenericPortlet.
  - You override/implement several methods inherited from GenericPortlet.
  - You use some supporting classes/interfaces
    - Many are analogous to their servlet equivalents
    - Some (portletsession) actually seem to be trivial wrappers around servlet equivalents in Pluto.

Some Terminology

Term	Definition
Portlet	Java code that manages a piece of web content and which may invoke services.
Portlet Container	Manages the lifecycle of the portlets (inits, invokes,destroys).
Portal	Displays the portal content provided by the container. The portal is responsible for the actual layout.
Portlet Application	A webapp containing a group of related portlets, content, supporting jars, etc.

The Infamous Big Picture

- As a portlet developer, the previous set of classes are all you normally touch.
- The portlet container (Pluto) is responsible for running your portlets.
  - Init, invoke methods, destroy.
- Portlets have a very limited way of interacting with the container.
  - It is a black box->black hole.
  - The API is basically one-way.



Some Generic Portlet Methods

Method	**Description**
Init	Called when the portlet is created. Override if you need to set initial params.
doView	Controls what happens immediately before the portlet is displayed in view mode. Normally you override this.
doHelp, doEdit	Other portlet display modes
processAction	Place for handling any <form> actions before turning over to the display mode method (like doView). You should override this for web forms.

Some Supporting Classes/Interfaces

Class	**Description**
PortletContext	Similar to servlet context; get context info and the RequestDispatcher from here.
PortletSession	Stores attribute information for a single portlet application across multiple requests.
RenderRequest, RenderResponse	The request and response objects available to the doView() method. Similar to the normal servlet request
ActionRequest,ActionResponse	The request and response objects available to the processAction() method. Similar to the servlet request and response objects.
PortletURL	Use this to create URLs that reference the portal.
PortletRequestDispatcher	Use this to include/forward to a JSP or servlet in the same portlet app.
WindowState	See if you are in minimized, maximized, normal state.

In Action: Get started.

```
public class JunkPortlet extends GenericPortlet {
    public void init(){
            //Do any initialization.
    }
    //Rest of the methods on following slides go here.
…
}
```

Override doView()

```
protected void doView( RenderRequest req, RenderResponse
  res)
  throws PortletException, IOException {
  //Include the desired JSP or HTML page.
  //We could also use out to write directly to the response.
  WindowState state=req.getWindowState();
  if(!state.equals(WindowState.MINIMIZED)) {
    res.setContentType("text/html");
    PortletRequestDispatcher rd=

      getPortletContext().getRequestDispatcher("MyJSP.jsp
  ");
    rd.include(req,res);
  }
  }
```

The JSP Page

```
<portlet:defineObjects/>
<%
PortletSession
    portletSession=renderRequest.getPortletSession();
portletSession.setAttribute("localattname","localattval");
PortletURL url=renderResponse.createActionURL();
String theActionString=url.toString();
%>
HTML Content is here.
A form is below.
<form method=post action="<%=theActionString%>">
<input type=…>
</form>
```

Some Notes

- Include the <%portlet:definedObjects/%> tag, which will instantiate renderRequest, renderResponse, and portletConfig objects.
  - You can then just use them, as with request, response, and other JSP implicit objects.
- The renderRequest gives you access to the PortletSession, if you want to store session variables.
  - One of the trouble points.
- The renderResponse gives you access to the PortletURL object.
- Use the PortletURL to generate a URL for the <form action>
  - So that it points to portlet container and gets handled by the processAction() method, rather than going of into space.
  - Handle href URLs similarly.
  - This is one of the sticking points.

Lastly, Override processAction()

- When you invoke the form on the previous JSP, the portlet container will pass the action handling to the processAction method.
- The ActionRequest can be used to get any of the <input> parameters in a way similar to the usual HttpServletRequest.
- When the processAction method returns, the container then invokes the appropriate do method (usually doView).
- If you need to pass <form> parameters on to doView, add them to the ActionResponse.
  - This will give them to the RenderRequest.
  - The example shows how to add ALL parameters.

```
public void processAction
   (ActionRequest request,
   ActionResponse
   actionResponse) throws
   PortletException,
   java.io.IOException {
   //Process request parameters
   …
   //Add any other request params
   // to the renderRequest
   actionResponse.setRenderPara
   meters(request.getParameterMa
   p());
}
```

Deploying Portlet Applications

- The portlet container (i.e. uPortal) runs as a distinct web application.
  - That is, it has its own directory in tomcat.
  - Moreover, it runs as a separate context, with its own classloader, session management, etc.
- Portlet applications are deployed as distinct war files/web applications.
  - You go through the container webapp to get to the portlet webapp.
  - Portlets in the same application share jars, classes, and runtime stuff like request and session variables.
  - Portlets in different portlet apps do not share anything.

A Critique of JSR 168

- There is no way to share data/objects between portlet applications.
  - So all grid portlets would have to be in the same portlet app.
- There is no way to extend the portlet API to add such services.
- There is a lack of general purpose portlets.
  - Write now, you make specific extensions to GenericPortlet for each portlet you develop.
- JSR 168's MVC approach is incompatible with Turbine, Struts, ….
  - The issue is managing <form action=""> and href URLs.
  - No fundamental problem, but this is a Gap that must be filled.
  - JSF Portlets, for example.
- Do defined way to integrate with portal-specific services (i.e. logins).
- No inter-portlet communication.
- (I conclude that portlet container developers will solve these problems but in container-specific/proprietary ways, so only trivial portlets will be truly interchangeable between containers).

JSR 168 Long Term Survival?

- There are many JSR 168 compatible portals out there.
  - GridSphere, uPortal, WebSphere, Jetspeed2, SunOne…
  - Most wrap Pluto to varying degrees.
- Given that many commercial vendors' previous portlet APIs were richer, and there is no standard way to extend JSR 168 to even address its shortcomings, it seems uncertain how much long term support/commitment beyond lip service they will provide.
  - What do container vendors really gain by interoperability any?

Strategies for OGCE

- Despite the criticisms, JSR 168 (and WSRP) are the only games in town for standard portlets.
- We are a "portlet vendor", not a container vendor.
  - We have a rich set of components already.
  - So concentrate on making OGCE portlets truly interoperable across containers.
    - Jetspeed2, uPortal, GridSphere.
- Adopt JSR 168 but make documented additions (as plugin tag libraries) to solve service problems stated earlier.
  - More detail later.
- Develop general purpose portlets (extensions to GenericPortlet) to simplify deployment of portlet types.
  - Developers can stick with their favorite (Java) webapp develop framework.
  - JSP, WSRP, Turbine/Velocity, Struts, JSF
  - Will require some standard practices.
- Use and advertise CoG portlet APIs.
- Work now on Grid security issues for WSRP

Installing uPortal

Precursor for Lab Exercises

Get the Code

- Get Tomcat 5.0.2x
- Get Maven 1.0 (for Marcus)
- Get ant from here: http://ant.apache.org/bindownload.cgi
- Go to http://www.uportal.org/download.html
- Download the uPortal-only version of 2.3.4
- Get Hypersonic from here
  - http://sourceforge.net/projects/hsqldb.
- Copy hsqldb.jar into uportal's lib/ directory.
- Open two windows.
- In first, start hypersonic like this:
  - java org.hsqldb.Server -database uPortal2 -port 8887
  - Your classpath should point to hsqldb.jar
  - Use java –cp lib/hsqldb.jar to do this.
- In second window, build uPortal. Just follow the instructions.
- Fire up tomcat in the second window, point your browser to http://localhost:8080/uPortal, and make sure you can log in (admin/admin)

Grid Portlet Examples

Marcus Christie

New COG APIs

Gregor von Laszewski

Implementing Portal Services

Portal “Services”

- “Service” means specifically a capability object in the portal that
  - Needs to be instantiated only once.
  - Needs to be shared between portlets (and stateless?).
  - Needs to exist independently of any portlet’s lifecycle.
  - Needs to be accessible by portlets from all webapps.
- Services implementations may be outside the portal, so the “portal service” is the internal client application in this case.
  - Still need to manage its creation, access, etc.
- Prototypical service: cross-portlet application data storage.
  - Particularly useful for storing GSSCredentials

Where Are the JSR 168 Service Managers?

- JSR 168 provides no facilities that allow you to
  - Configure available services and their implementations.
  - Control their instantiation
  - Get service instances in your portlets
- More serious: JSR 168 does not provide you with extension points to add these necessary services.
- Note the containers (like Pluto) do have this (for logging, for example) but there is no way for a portlet to get services from its container.

Solutions for Sharing Data

- Motivating problem is sharing GSSCredential object between portlet apps.
- For testing, this can be done simply by using singleton HashTables/HashMaps
  - Let’s call it MyStore.
  - But need to implement a clean up procedure for cleaning up old credentials.
- Two problems:
  - You now need to get an instance of MyStore in your portlet.
  - You really want a more general solution to support this as a remote service (anticipating WSRP).
- Would Spring be useful here?

How Do I Access MyStore?

- Basic solutions:
  - Use all static methods
    - Static Object MyStore.getThing(String id);
    - Can’t convert into a service.
  - Just create it and let the constructor check to see if static store has been created.
  - Write a factory to do it instead of calling the constructor.
    - Useful way of creating more than one service.
  - Use Tomcat JNDI built in factories.
- The last requires some explanation.

Using Tomcat JNDI

- Tomcat comes with a built-in JNDI implementation.
- You can use this to store and share objects (i.e. service handles) across Tomcat using a fairly standard API.
- Say for example you have implemented a service for storing credentials, MyStore. It stores them in a hashtable or a database or whatever.
- You now need to get these credentials. First you must get the MyStore object in your portal.

Configuring Tomcat JNDI for Global Objects

(I have more extensive notes. This is basic procedure).

1. Place your service jar or class in tomcat's share directory.
2. Add the resource to server.xml's <GlobalNamingResources>
    1. Specify a factory (Bean, DatabaseFactory, etc.)
3. Add your portlet webapp specifically to the server.xml <context> list.
4. In your portlet's <context>, add a <ResourceLink> that points to the GlobalResource.
5. Also modify your portlet's web.xml to specify the resource.

What Have You Gotten For All That?

- You can now get an instance of MyStore from JNDI using the procedure on the right.
- The Context objects are part of JNDI.
- Get an instance of the MyStore object

```
<%@ page
    import="javax.naming.*"%>
<%@ page
    import="mep.junk.MyStore"%>
<%
Context initCtx=new
    InitialContext();
Context
    envCtx=(Context)initCtx.looku
    p("java:comp/env");
MyStore myStore
    =(MyStore)envCtx.lookup("bea
    n/MEPBeanFactory1");
%>
```

Problems with Tomcat JNDI

- It only works with Tomcat and Tomcat-based implementations.
- It really only adds value in a few cases:
  - You want to use Tomcat’s DB connection pooling.
  - Your object requires a lot of initialization (like JavaMail).
- It really requires a lot of tomcat configuration, which I view as fragile.
  - If you tamper with server.xml or web.xml, you may screw it up.
  - If you download a new version of tomcat and copy over the old webapps but forget to change server.xml, things break.

In the End, It Should Be Just Part of a Tag Library

- How ever you create the MyStore object, this should be hidden.
- We need a set of OGCE defined object tags that instantiate these things.
- You then use it similarly to a JSP implicit object (request, response, session).

```
<%
<portlets:definedObjects/>
<ogce:definedObjects/>
PortletSession
    portletSession=renderReq
    uest.getPortletSession();
myStore.setAttribute(portletS
    ession.getId(),"Hollow
    World");
%>
```

Other Services to Put in <ogce:definedObjects/>

- We need a list:
  - Session storage
  - Persistent storage
  - Metadata/context
  - Logging
  - COG objects?
- Note again these may be clients to remote services.

Supporting Developer Frameworks

Struts, turbine, etc.

Supporting Development Frameworks

- The JSR 168 model imposes specific MVC/Model 2 implementation.
  - doView does the view
  - processAction handles the action.
- This does not map to most pre-existing web page development methodologies, which define their own MVC implementations.
  - Struts has the action servlet.
  - Turbine/velocity does its thing.
  - JSF is the exception, but still new.
- The issue is in the <form> action URL, which must be pointed to the portal container, not the Struts do action servlet or the turbine servlet or whatever.
- We will not convince Struts developers (for example) to rewrite all of their Struts pages to comply with JSR 168.
  - Instead, we need a solution for supporting these other frameworks in portlets.
- I don’t feel these are fundamental problems, but we just need a bridging solution between the two.

A Related Issue: Portlet Types and Templates

- Even if you have JSP pages, you must still write a lot of GenericPortlet extension classes to handle the actions.
- Making a lot of specialized extensions of GenericPortlet for each application is not a good approach.
- Consider instead the old Jetspeed model
  - You don’t usually develop portlet code, you develop templates (JSP, Velocity, etc) for your content.
  - Various general purpose portlets (JSPPortlet, VelocityPortlet) exist to handle these templates.

OGCE Portlet Types

- I propose the we follow this template/handler development model.
  - Provide general purpose portlets that support legacy applications with a few modifications.
- Why do this? We need to simplify the lives of science portal developers.
  - They may have their own preferred development tools already
  - Need to provide a simple way for them to develop their own portlets.
- Suggested portlet types
  - OGCEJSPPortlet
  - OGCETurbinePortlet
  - OGCEStrutsPortlet
  - OGCEJSFPortlet
  - OGCEWSRPPortlet

An Approach to OGCEJSPPortlet

- The main problem is that JSR168 portlets require you to handle your actions in the processAction() method.
- For a vanilla set of JSP pages, change the form action to the PortletURL (see earlier slides).
  - Add the original action URL as a hidden parameter with a name, like ogce.next.page
- OGCEJSPPortlet's processAction() just checks for this parameter and then resends all parameters to doView.
- doView loads the next.page with the request dispatcher.
  - Ogce.next.page may be a jsp, servlet, etc.
- Within certain restrictions, this should work.

Approaches to OGCEVelocityPortlet

- (Eric/Marcus to lead)
- These are much more complicated, I think.
- What is the Sakai/Chef solution?
- What is Jetspeed’s solution?

Summary of Proposed Developments

- Grid portlets that run in any JSR168 container.
- An OGCE tag library of objects for handling basic services.
  - Local store and persistent store are two.
- OGCE Portlet sets for supporting development frameworks.
  - Start with JSP, Turbine (and examine JSF).
- Secure WSRP suitable for Grid Portlets.

Core SC Activities

- Develop basic 168 grid portlet suite with necessary services.
  - Demonstrate working in uPortal, Jetspeed2, GridSphere.
    - Can "home" in uPortal, especially if you are developing specific portals, but I think interoperability is important for OGCE.
  - Use CoG 4.
  - Service data sharing strategy (need to finalize now)
  - Adopt JSF portlets now?
- Other JSR168 portlets as described in the SOW.
- Need WSRP/JSR168 portlet
- Turbine/Velocity support in JSR 168 for CHEF.

Longer Term Activities

- More persistent data service support
  - Gregor and Mary listed several, I think.
- WS-Context as an approach for data sharing.
- WS versions of COG interfaces
- Grid-enabled WSRP
  - That is, SOAP security, secure conversation, ability to transfer credentials.

Who does what?

- Grid portlets with CoG 4:
  - Maven packaging
  - Filebrowser/Gridftp: Nythia (GridFTP), Jonathon, (gridftp)
  - GRAM: Mike, Gopi,
  - GPIR: Eric, Akhil
  - GridContext (Liang)
  - Workflow portlet (Mike, later)
  - Condor ()
  - OGSA-DAI portlet
- Sakai hacking: Make progress in getting metadata repository into the Sakai.
  - Joe
- JSF Portlets in uPortal: test case
  - Ahmet, Mehmet, Greg
- API for Data Sharing:
  - Sunghoon, Chuck, Marcus, **Marlon**
- Turbine/Velocity portlet:
  - Marcus, Eric
- WSRP
- OGCE capability portlet

uPortal installation: this was done in a room full of people who know this general stuff. It was still very painful.

1. In general, the installation of "uportal-only" was very fragile—had to follow instructions carefully. A couple of recommendations:
    a. Document what happens when you do things incorrectly. Didn't start the database? You will get this error. Database is not correct? This is what happens. uPortal takes forever to load? This is what is wrong.
2. There were too many different versions of the instructions, not all of which agree. The readme in the installation was mostly correct but had wrong/missing instructions for dealing with the hsqldb.jar and starting the database independently.
3. When using uportal-only, you must remember to update the hsqldb.jar in the uportal/lib. Otherwise, it does not work. Why not update the hsqldb.jar in the uportal distribution to be the "correct" one.
4. Installing the portlet has a big potential error: you have to use "war-file-name.portlet-name-from-xml." If you screw this part up, your 168 portlets will not work. While this is documented, I personally find it hard to remember (out of all the possible name combinations) which one is correct. This really needs to be improved/automated.
5. Many found the default navigations non-intuitive.
6. uPortal errors for 168 portlets were not very detailed in the display. You have to go to the logs to debug.
7. A lot of people (who know ant) just ran "ant" with no targets, assuming that the default target was correct. This "worked" in that you got no errors, but the portal could not be started. This caused frustration and confusion. The readme has the correct targets, but why not just make these the default targets? Or, why not make the default target a "readme/help" that tells you what to do but does not itself do anything.

Some uPortal Questions

1. When is 3.0 coming out? What will be the changes from the current version?
2. What is the plan for uPortal 3.0 w.r.t. JSR 168? We understand that this involves big changes under the covers (channels->portlets), but what (operationally) will be the differences we will see as 168 portlet developers?
3. Will uportal 3.0 support WSRP clients that are 168 portlets? Or will this use the channel adapter from the uportal 2.3 base?
4. Where can I get more skins? The defaults are not pleasing.

NMI OGCE YEAR 2 SOW

Deliverables Plan

We will once again schedule two major releases, one for Supercomputing 2004 and one for May 1, 2005. We will have supplemental software releases as necessary. Our primary technical focuses this year will be a) porting Year 1 portlet software to the new portlet standard, JSR 168, b) developing and implementing common Grid portlet APIs, and c) developing advanced portlet capabilities, listed below. Efforts a) and b) will be group deliverables, while c) will mostly involve efforts at individual institutions.

The OGCE is intended to serve as an open consortium that can include external contributions. During the second year, we will work closely with selected communities to integrate externally developed tools and (more importantly) specify the contribution/integration process for third party software. Specific deliverables will include the integration of externally developed portal capabilities from NEESGrid, CMCS, and the Fusion Grid Portal projects into the OGCE release. We will also work with these groups to forward port their software to JSR 168 compatible versions. We also anticipate supporting many additional, highly visible NSF portal efforts, particularly TeraGrid and the LEAD portal.

Programmatic deliverables for this year will be closer integration of the portal build and test process with the GRIDS center testbed. This integration will involve automated build and report systems, but also we will develop (using HTTPUnit) an extensive set of portal tests that will be able to verify both the functionality of the portal and also the underlying Grid software.

Supporting Collaborative Communities

1. We will continue to lead the TeraGrid User Portal design and development efforts. Mary Thomas is the lead designer, with assistance from IU and ANL team members. An initial demonstration portal will be shown at Supercomputing, with upgrades through the year. A detailed User Portal Plan document has been written and is available.
2. We will integrate selected NEESGrid and CMCS capabilities into the OGCE release. We will work with these two groups to adopt additional OGCE capabilities.
3. We will participate in the DOE SciDAC program’s Portals Consortium. This is a newly formed, volunteer consortium of DOE project PIs from TACC, IU, ANL, and Lawrence Berkley National Lab. This consortium’s goal is to coordinate portal develop efforts within the SciDAC program, using the OGCE portal release as a baseline.

Software Deliverables

Global Deliverables

These are team efforts and will involve all OGCE team members.

1. Migration portlets to support JSR 168 container compatibility. We will demonstrate, and have an initial release, of JSR 168 compatible portlets available for SC2004. These will focus on Grid portlets (credential management, GridFTP, GRAM, etc.)
2. A fully JSR 168 compatible release will be available in the May 2005 release. This will include all major OGCE portlets. We will demonstrate container independence: our portlet releases will be able to run in most/all major JSR 168 containers.
3. We will identify and implement necessary portal container services for implementing fully interoperable JSR 168 grid portlets. JSR 168 portlets have a standard API that can be invoked by the container, but there is not yet a standard API for the reverse operation (invoking services provided by the portal container from within a portlet). We will work with the appropriate communities (primarily the Sakai and GridSphere developers) to identify these services.
4. Portal support for GT4 services and WSRF. This will be delivered in May 2005.
5. We will develop and implement a generic Grid Portal API for all major portal foundation services: authentication, file transfer/management, persistence, etc. This will simplify the development of third party Grid portlets that can be plugged into the OGCE framework. We will work through the Global Grid Forum's SAGA Research Group to make sure that our efforts will be compatible with the larger Grid application development community. This effort will be lead by ANL and TACC, but will involve all participants.

Local Deliverables: IU

1. We will deliver portlets and services for audio/video collaboration. These portlets will include AV session management portlets as well as AV clients. We will include portlets and services that can be used to participate in Access Grid sessions over unicast (rather than multicast) networks. We will also include portlet clients capable of connecting to H.323/Polycom sessions. This will be included in the Supercomputing release.
2. We will develop general purpose portlets and services to support Geographic Information Systems. GIS clients and services are widely used by government agencies for planning and emergency preparedness/homeland defense, so these capabilities will broaden the OGCE portal appeal. For Supercomputing 2004, we will develop versions of GIS Web Feature and Web Map services, along with Web Map clients. For the May release, we will include additional GIS services.
3. We will develop a secure version of the Application Factory Service that can be used to wrap legacy science applications in launch them through the portal. This will be integrated with the NCSA OGRE system and (through the general WS-Notification interface) the IU NaradaBrokering system. These will be initially demonstrated at Supercomputing 2004, with upgrades for the May release.
4. We will develop data grid portlets and services based on the OGSA-DAI or similar services. This will include an initial release for SC2004, with upgrades for the May release.

Local Deliverables: TACC

1. Infrastructure:
    a. Development of OGCE Interfaces and Patterns (Working directly with Argonne)
    b. GridPort Toolkit implementing OGCE Interfaces and supporting OGCE API
2. Portlets and services:
    a. Storage Resource Broker (SRB): working with SDSC team to create portlets and web services for SRB.
    b. TACC User Portal (TUP) bundle: will deliver portlets used for a HotPage portal
    c. GPIR: additional features and capabilities including ability to store key CoG objects (Task, Execution, etc.). TACC working with Argonne on this deliverable
    d. Portal account creation and management portlets, including session to session persistence via GPIR
    e. Pegasus Workflow Portlets – TACC working with ISI to integrate Pegasus into GP3 and OGCE
3. Portals (in progress or planned):
    a. TeraGrid User Portal – TACC and IU collaborating
    b. TACC Molecular Dynamics Portal: will deliver portlets for the following MD applications: NAMD, AMBER, CHARMM, and GROMOCS
    c. DOE Fusion portal
    d. CMCS Portal (PNNL): TACC working with Argonne (PI) to support DOE portal development

Local Deliverables: UMich

1. Infrastructure
    - Liaison with Sakai Project and uPortal Projects
    - Continued support for KX.509 as a single sign on mechanism
    - Investigate the integration of Shibboleth, the Grid, and the portal
    - Integrate Grid into Sakai release
2. Portlets and Services
    - Accounting and job submission portlets
3. Portals (in progress or planned)
    - NEESGrid Portal - Convert to OGCE Current version once it is released. Available for SC04

Local Deliverables: University of Chicago

A. First Quarter FY2

1. Adapt the deliverables of the project dependent on the timelines and deliverables used in future Globus Toolkit releases with the focus on GT4
2. Assist in the move of the Web site to Indiana University
3. Assist in the move of the Code repository to Indiana University.

4. Provide a direct port of Condor services within the Java CoG Kit allowing the development of Condor style job submissions directly through the Java CoG Kit.
5. Improve the Workflow portlet backend service
6. Provide a port of the Java CoG Kit for GT4
7. Continue the support of GT2.4
8. Support for future Globus toolkit versions

B. Second Quarter FY2

1. Development of a File Access mechanism as part of the Java CoG Kit.
2. Development of a Portlet to conduct File access based on item 1.
3. Development of a prototype of a Grid Command manager that an be started through Webstart within the portal and is used to mange many user controlled jobs.
4. Support for future Globus toolkit versions

C. Third Quarter FY2

1. Improvement of the workflow services while using the newest service oriented Grid technologies.
2. Support of WebDAV in the File access component as part of the Java CoG Kit.
3. Development of a Portlet to the Simple CA to allow other communities to more easily run their oown certificate authority.

D. Fourth Quarter FY2

1. Showcase the functioning workflow service while using existing Globus and Condor hosting environments.
2. Investigate the possible integration of smart cards and other external devices as part of the security integration within the portal.
3. Assist the GGF SAGA working group in formulating documents for a simplified API to the Grid.
4. Support for future Globus toolkit versions

Outreach Deliverables

We will continue to participate in the major Grid community

1. Demonstrations at Supercomputing 2004 to coincide with release.
2. Presentations at GlobusWorld
3. Participation in Global Grid Forums

We will also organize a Grid Portal tutorial.