**LEAD Portal and Security Architecture Document**

**LEAD Portal Architecture**

The LEAD Portal is, essentially, a set of Java portlets complying with the JSR 168 Java Portlet 1.0 specification hosted inside GridSphere which provides a JSR 168 compliant portlet container. For the user, the LEAD Portal takes care of loading the user's security credential and making it available to the LEAD portlets. The portlets in turn use the user's security credentials to interact with LEAD web services on the user's behalf.

The LEAD Portlets have been developed using a couple of different view technologies. For historical reasons, there are some portlets that use the Velocity templating engine for defining the view. More recently, the LEAD portlets have been implemented using JavaServer Faces (JSF), a stateful, component-oriented web framework. The MyFaces core JSF implementation library is used as well as the Tomahawk add-on JSF component library. The GeoGUI also uses JSF as well as the Yahoo User Interface (YUI) library which is a JavaScript utility library. There are many different web frameworks that could be used to build Java portlets and no specific one is required. On the backend, the LEAD portlets primarily use the XSUL (Web and XML Services Utility Library) client libraries to interact with LEAD web services.

LEAD Portal account and credential creation is accomplished via the PURSe portlets. The PURSe portlets are JavaServer Faces portlets that build on top of the PURSe API (PURSe stands for "Portal-based User Registration Service"), which takes care of generating a user's security credential from a Globus simpleCA installation and storing that credential in a MyProxy server. The PURSe portlets are the only part of the LEAD Portal that require a local installation of Globus. The Globus installation is a minimal one, requiring only some basic security libraries and a persistent MyProxy server. PURSe then requires the following to be configured:

- A database (using MySQL)
- File paths where the Globus installation is located
- A MyProxy server in which to store generated credentials
- A simpleCA configured to generate credentials for LEAD portal users

The PURSe Portlets have a pluggable registration module API. There are two registration modules that are relevant to the LEAD Portal: the MyLEAD and the GridSphere registration modules. The MyLEAD registration module takes care of contacting the MyLEAD Agent and requesting that an account be created for a user using the identity generated for that user. The GridSphere registration module takes care of creating the user's GridSphere portal account.

MyProxy single sign-on is achieved by creating a GridSphere authentication module that will attempt to log into the MyProxy server configured with PURSe to retrieve a proxy credential using the supplied username and password of the user. If successful, the user's proxy is loaded into the user's portal session and is then available to other portlets via the ProxyManager API, which simply takes the username of the user and looks up the user's proxy credential in that user's portal session.

The LEAD Portal, as mentioned above, runs as a set of portlets in the GridSphere portlet container. GridSphere and the set of LEAD portlets run as web applications inside of Apache Tomcat, a Java servlet container (currently version 5.5.x). Tomcat requires some special configuration to work with GridSphere and the portlets.

- The Tomcat Manager application needs to be installed to allow GridSphere to start/stop/reload portlets
- The main Connector needs to be configured so that the session cookie uses an empty path, i.e., "/", to allow the portal container (in this case, GridSphere) and the portlets access to the same session.
- Some grid security libraries as well as the ProxyManager API is required in the shared
classloader (i.e., these jars are needed in the shared/lib directory).

Apache HTTP Server is used as the front end to the LEAD Portal and is installed on the same server as
the LEAD Portal. It is configured to handle SSL security and proxies requests to Tomcat.

The LEAD Portal uses version 2.1.5 of GridSphere. GridSphere requires the following configuration.

- GridSphere must be configured to allow it to access the Tomcat Manager application, as
  referenced above.
- GridSphere requires a database server and currently we are using MySQL version 5.0.21. (Note:
  since GridSphere uses Hibernate as its persistence layer, there are several different options for
  which database server to configure for GridSphere)
- The MyProxy single sign-on authentication module needs to be configured with GridSphere
- There are various other modifications to the layout of the portal and the definition of various
  tabs and sub-tabs, including a LEAD "theme".

The GeoGUI relies on MapServer for generating projections of geospatial data on maps and overlays
this on the Google Maps interface. Currently, MapServer is used to provide positions of radar stations.
The use of Google Maps requires an Google Maps API key, which is obtained for a particular domain and
path.

**LEAD Portlets**

The following LEAD portlets are briefly described in terms of their key functionality and their
dependencies.

**Experiment Builder Portlet**

The Experiment Builder portlet is a Velocity (a Java templating framework) based portlet that provides
the user with the ability to create, configure, launch and monitor experiments. The dependencies of the
Experiment Builder portlet include:

- **MyLEAD Agent** - for retrieving the user's projects and experiments, and for creating new
  experiments in the user's workspace.
- **GPEL Workflow Engine** - for retrieving workflow instance status and workflow templates, and for
  creating new workflow instances
- **Data Catalog** - for searching public data for workflow inputs
- **XBaya** - needs the URL where the XBaya jars are deployed when constructing a JNLP file to
  launch XBaya
- **Auditing Service** - registers new workflows with
- **Service URLs** - passes several other service URLs in the LEAD context header to the workflow
- **GeoGUI** - uses a special instance of the GeoGUI for configuring the model domain

Since the Experiment Builder is the primary starting point in the launching of a workflow, it sets the
experiment ID which is also the notification topic for messages sent related to the workflow. It also
sets the LEAD context header including the URLs of the various LEAD services required in the execution
of the workflow.

**MyWorkspace Browser Portlet**

The MyWorkspace Browser portlet is a JavaServer Faces portlet that provides browsing and search
capabilities over a user's personal MyLEAD space. The dependencies of the MyWorkspace Browser
portlet include:
• MyLEAD Agent - for retrieving the user's workspace and also for some updates to the user's workspace like deletion of nodes and updating attributes
• MyLEAD Server - the search interface talks directly to the MyLEAD Server
• MyLEAD Publisher - for publishing user's workspace artifacts to the Data Catalog service
• Thredds Data Server overlay of the MyLEAD repository - TDS access to the MyLEAD repository is used to get OPeNDAP and file download access to files (for IDV visualization and downloading, respectively)

**PURSe Portlets**

The PURSe Portlets are a suite of JavaServer Faces portlets that provide registration and identity credential generation capabilities. The PURSe Portlets and their dependencies are described above.

**Data Search (GeoGUI) Portlet**

The Data Search (GeoGUI) portlet is not a portlet per se but rather a standalone web application that is integrated into the LEAD Portal via an iframe (an embedded HTML frame). As noted above, the GeoGUI has dependencies on MapServer and the Google Maps API.

**Security Architecture**

The security architecture relies on the public/private key cryptography available via the X.509 certificate model. All users and most services are expected to have a X.509 credential. As discussed above, user's obtain a security credential from the PURSe API when they are approved for a LEAD account. Service credentials are generated for hosts on which LEAD services will run and are used by the LEAD services run on those hosts to securely interact with other LEAD services.

A distinction should be made regarding the kind of credential generated for users, since they resemble grid credentials and the only real difference is the way they are used within the LEAD services. These credentials are referred to as "identity credentials" because their only purpose is to serve to securely identify the user within the LEAD architecture. This distinguishes them from grid credentials which also serve to authorize grid users for various grid activities.

The security model between the LEAD Portal and the various services is what we have referred to as the "trusted services" model. In this security model, the credentials for the LEAD services are generated from a single Certificate Authority (CA). The LEAD services then trust that CA and hence each other. If a request comes in from a service with a security credential signed by a different CA or without a security credential, then that request would be automatically rejected. The LEAD services also recognize the CA that is used to generate the LEAD Portal users' credentials and will trust such requests and perform actions on a user's behalf. The web services security is implemented by tunneling Web Services operations over SSL (XSUL implements this with the Java library PureTLS).

Fine grained authorizations are accomplished via an access control list mechanism provided by the XRegistry service. The XRegistry service allows groups of users to be defined and authorizations to be granted to users and/or groups. Authorizations can be set on application services and workflows.

Auditing and accounting is performed by the Auditing service. When a user launches a workflow from the LEAD Portal, the LEAD Portal contacts the Auditing service and registers the workflow with it. The Auditing service then listens for special auditing notifications sent by applications services and builds up a history of what jobs were run during the execution of the workflow. The Auditing service then contacts accounting databases in TeraGrid to look up how many service units (SUs) were consumed by a particular job, and by summing these up it provides the total SUs consumed by a particular workflow. In the near future, the Auditing service will also provide the ability for an administrator to set certain constaints on a user's account like a quota or sub-allocation of SUs or a restriction to only be able to run a certain number of workflows within a particular timeframe. It will be incumbent on the application services to check with the Auditing service to see if the user has exceeded these constraints and then disallow the invocation of that service.
A service was developed to handle a few different scenarios where the need for a delegated, renewed credential is desired, and this service is called the Session Credential Management Service (SCMS). SCMS provides LEAD services with the ability to easily obtain a Dr. LEAD proxy credential. This is a special grid identity/credential under which all grid operations of the LEAD services are executed. This identity maps to the LEAD Community Allocation on TeraGrid, and hence the Dr. LEAD account is a TeraGrid Science Gateways "community user account". In the near future, SCMS will also provide a means by which a user can delegate his/her credential to a service that will return a token that can be given to LEAD services (via a SOAP header parameter) that would allow that service to retrieve a copy of that user's credential. The user can destroy this "session credential" as needed, else it will be terminated after a period of inactivity. This "session credential" can thus be delegated to a workflow to run on a user's behalf and under the user's own identity on the grid.