

LEAD Portal Requirements

v. 1.0-DRAFT

Marcus Christie*
Suresh Marru†

January 28, 2005

Contents

1	Introduction	2
1.1	Overview	2
1.2	Purpose	3
1.3	Terminology, Assumptions and Acronyms	4
1.3.1	Terminology	4
1.3.2	Assumptions	6
1.3.3	Acronyms	7
1.3.4	External References	7
1.4	LEAD Portal Architecture	7
2	LEAD Portal User Groups	8
2.1	LEAD Learning Communities	8
2.2	Requirements for Education Users	11
2.3	Template Use Case Scenario	12
3	Portal Framework Functional Requirements	13
3.1	Introduction	13
3.2	General Requirements	13

*Email: machrist@cs.indiana.edu

†Email: smarru@cs.indiana.edu

3.3	User Management	13
3.4	Group Management	13
3.5	Security	14
3.6	Customizability	14
3.7	Inter-Portlet Communication	14
4	Portlet Functional Requirements	14
4.1	Experiment Builder (MyExperiments)	14
4.2	Workflow Composer	17
4.3	Workflow Browser	18
4.4	MyWorkspace (MyLEAD Browser)	19
4.5	Registry Catalog	20
4.6	Resource Management (User Preferences)	21
4.7	Account Management	21
4.8	Application Code Management	21
4.9	Other Tools	22
	4.9.1 Visualization Services	22
4.10	Not Touched On	22
5	User Interface Design Requirements	23
5.1	Personalization	23
	5.1.1 Overall Look and Feel	23
	5.1.2 Flexibility in Design	23
	5.1.3 Portal Interface and User Groups	23
5.2	Minimum Computer/Browser Requirements	24

1 Introduction

1.1 Overview

The Web-based LEAD Portal is the principal point of entry into the LEAD environment for individual users. Consequently, the Portal must be a self-contained system for understanding, using, and helping improve LEAD. It is viewed as a central component of LEAD because it will

- provide a single, customizable interface to the world of mesoscale meteorology research and education and thus simplify the user's ability to interact with a complex environment consisting of streaming and archived

data, distributed resources, advanced tools, remotely-controllable observing systems, and continually changing data formats and policies,

- allow users to create and manage their own work space that can be used to publish data for limited or open accessibility,
- function as a “real time” or “live” system, providing continuously updated information on the status of grid resources, current weather, observing systems and data feeds, and user-initiated activities (e.g., running jobs).

This document specifies what is required for the LEAD portal. We begin by establishing a common base for later discussion by defining common terminology and acronyms and explicitly stating assumptions. We specify the different groups of users for whom the LEAD portal will be built as well as the use cases the portal should support for each user group. We then address the requirements of the framework on which the LEAD portal will be built. The requirements of the “portlets”, the individual units of functionality within the portal, are then enumerated. Finally we look at user interface requirements such as color schemes and portal layout.

1.2 Purpose

The purpose of this document is to express the vision for the LEAD portal. This document describes what the LEAD portal should do. However, we do not specify here how the LEAD portal should be implemented. That job is left for the LEAD Portal Implementation Plan which will be explicit about what work will be done, when it will be done, and how it will be done.

A few words should be said about the relationship between the LEAD Portal Requirements document and the LEAD Implementation Plans. We say LEAD Implementation Plans since with each generation of the LEAD Portal (a new generation is expected to come about approximately every six months or so), there will be a new LEAD Portal Implementation Plan to guide the development of that generation. However, there may or may not be a newly revised LEAD Portal Requirements document. It’s possible (though unlikely) that this document will not see a revision for the entire 5 years of the LEAD project, in which case the LEAD Implementation Plans would build on one another to asymptotically approach the vision of the LEAD Portal that is laid out here. The point is that the LEAD Portal

Requirements document specifies where we want to eventually arrive with the LEAD Portal whereas the LEAD Portal Implementation Plan specifies what work must be done next to arrive there.

Another purpose of this document is to show linkages with other requirements and specification documents within LEAD.

1.3 Terminology, Assumptions and Acronyms

This section lays out a common set of terms, assumptions and acronyms used throughout this document.

1.3.1 Terminology

Portal A website with the following characteristics:

- Users have an account on the portal, and the portal associates various user-specified settings with that user account
- The portal aggregates web content and applications across a particular domain
- The user authenticates to the portal only, and doesn't need to authenticate to any services accessed through the portal (i.e., single sign-on).

Portlet A component within a portal that serves up a particular type of content or that provides an application to the user.

Web Start Applications Java applications that are launched by a user clicking on a link on a web page. The application is downloaded, installed, and started within the user's desktop environment automatically.

Applets Java applications that are loaded as objects embedded within the web page.

Services While a broad term, when used in this document it refers to a Web service, as defined by the W3C:

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network.

It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Services Provider A user or facility that hosts a service.

Investigation A collection of experiments having a particular purpose in mind.

Experiment The set of tasks performed to study current and future weather by visualizing current observational data or performing a forecast and analyzing the resultant processed data.

Resource Hardware necessary for the execution of an experiment. Examples include: compute clusters, long term data storage, scratch data space, services.

Model A computer science application, which provides an in-depth analysis of a phenomenon related to a specific domain, by solving set of mathematical equations.

Ensemble Forecasts Ensemble Forecasting is a method of increasing the accuracy of the forecasts. There are different ways of performing ensemble forecasts.

1. Simulating a particular model with different initial conditions.
2. Simulations with changing physical parameters for one set of initial conditions.
3. Simulations with variation of both initial conditions and physical parameters.
4. Simulation a forecast with different forecasting models like WRF, ARPS, MMF with same data, physics and initial conditions

Ensemble Analysis Ensemble analysis is a method of collecting the data from all the ensemble runs and rerunning the model with new data

Observational Data Meteorological information about the actual weather conditions provided by observational instruments like radars, satellites, radiosondes (weather balloons), surface aviation e.t.c

Output Data The data generated by a model after processing

Workspace The user's initial view of the portal after logging in, providing a view of the user's investigations and associated data.

Private Space Results of experiments and investigations that the user has not made public.

Public Space Results of experiments and investigations that the user has made public.

Model Initialization Setting up the initial observational data for the model to perform numerical prediction

Data Assimilation/Analysis Data assimilation or analysis is a method of interpolating all the observations to the forecasting grid

Forecast A term used to predict the future weather based on the current prevailing conditions.

Visualization A graphical way of representing the data

1.3.2 Assumptions

Model Executables The portal will assume the service provider to have control of the model source code and the executable are pre-built and ready to execute on the targeted compute servers

Simulations Scripts The service will have the knowledge of simulating the application, so a service will generate the simulation scripts on the server

Error and Failure Handling The portal will rerun the components if there is any error from the service.

1.3.3 Acronyms

ARPS Advanced Regional Prediction System is a regional to storm-scale atmospheric prediction system that includes a real-time data analysis and assimilation system, a forward prediction model, and pre and post forecast analysis components.

ADAS ARPS Data Assimilation System interpolates the observational data on to the ARPS grid by combining observations with a background field

WRF Weather Research and Forecasting Model is a non-hydrostatic, limited area model for storm, mesoscale and synoptic weather prediction. The WRF software can be applied to study convection, baroclinic waves, boundary layer turbulence, as well as data assimilation impacts, real-time weather phenomena and coupled-model applications

IDV Integrated Data Viewer is a Java-based software framework for analyzing and visualizing geo-science data

ESML Earth Science Markup Language is an interchange technology that enables structural and semantic data interoperability with applications without enforcing a standard format within the Earth science community

ADaM Algorithm Development and Mining system is a data mining toolkit designed for use with scientific and image data. It includes pattern recognition, image processing, optimization, and association rule mining capabilities

1.3.4 External References

IP-UFR Stands for *Implementation Plan-User Functional Requirements*, and refers to the User Functional Requirements in the LEAD Implementation Plan. For example, IP-UFR-Or6 refers to the Or6 User Functional Requirement (where Or6 is the sixth Orchestration User Functional Requirement).

1.4 LEAD Portal Architecture

How does the LEAD Portal fit into the broader LEAD system architecture? By and large, the LEAD portal is a client of services offered in the LEAD

system. Refer to figure 1 for a visual description of this setup. Note that this diagram is descriptive and not prescriptive.

In figure 1, we have the three core building blocks upon which the LEAD portal is built. At the foundation is the operating system and the Java virtual machine. As an example of this, we can consider Linux and the JVM provided for Linux systems by Sun. On top of this runs the servlet container, for example, the Apache Foundation's Jakarta Tomcat. Finally, on top of the servlet container runs the portal framework, an example of which is uPortal from JA-SIG.

The portal framework is able to host and manage portlet applications by means of a portlet container which is embedded in the portal framework. Following with our example, Apache Foundation's Pluto is the portlet container used in uPortal. The LEAD portlets then sit within the lead portlet container. The interaction of the LEAD portlets with LEAD services is via the client interface exposed by those services, compliant with the LEAD basic profile (WS-I+LEAD). To give an example of such interaction, the Experiment Builder portlet

- retrieves experiment information and workflow templates from MyLEAD; saves new experiments to MyLEAD,
- retrieves current notifications from the notifications broker,
- searches for data sources and stores in the Registry Catalog,
- launches an a workflow by submitting it to the Workflow Engine.

2 LEAD Portal User Groups

LEAD is targeted principally toward the meteorological higher education and operations research communities. LEAD also is developing learning communities, centered around teacher-partners and alliances with educational institutions, to bring the benefits of LEAD technologies to grades 6-12 as discussed in the following section.

2.1 LEAD Learning Communities

The LEAD Learning Communities comprises of a collaborative network of teachers, researchers, and students who interact to address a variety of issues

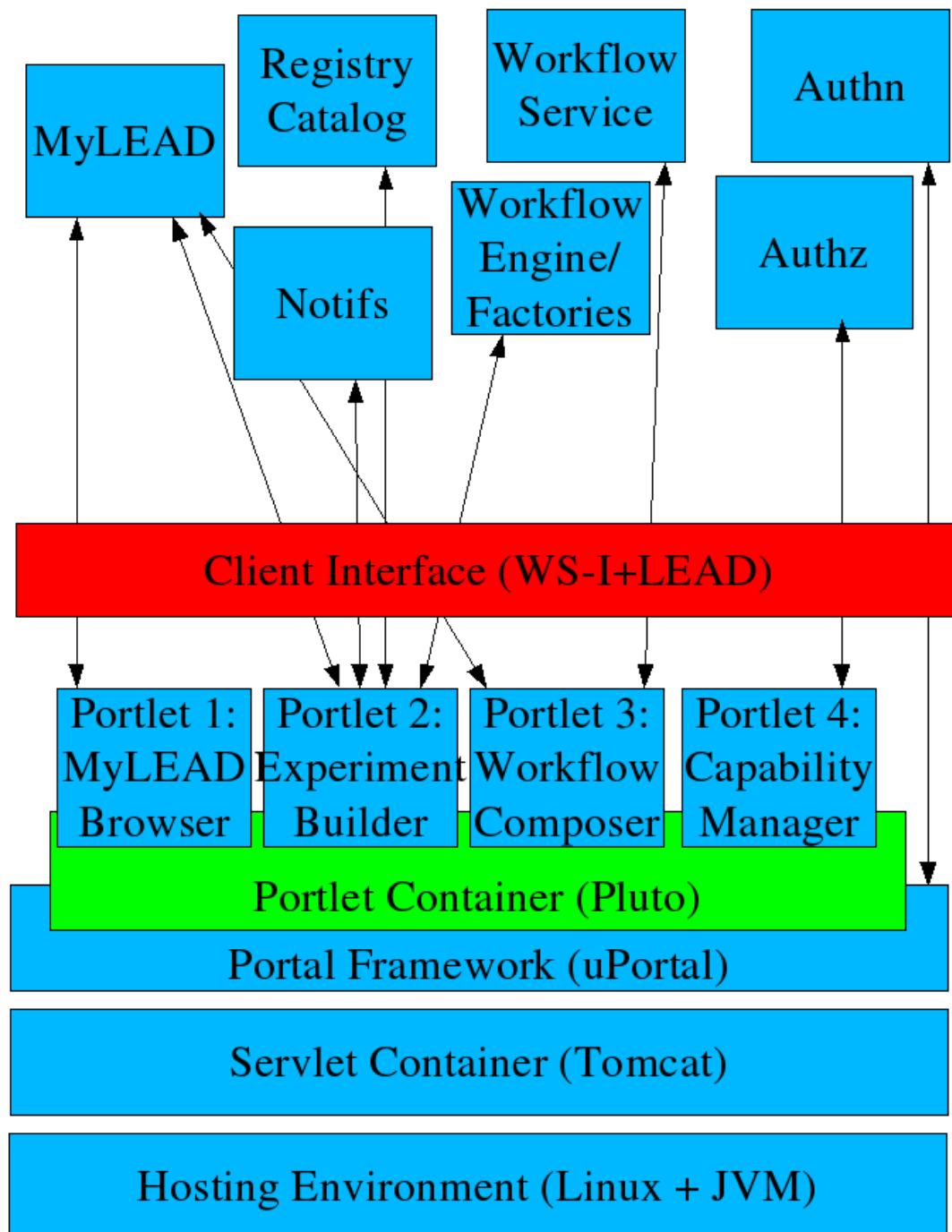


Figure 1: LEAD Portal Architecture

related to LEAD that are common to their community.

The Learning Communities are designed to enable and foster a two-way interaction between the LEAD developers and LEAD users to maximize the overall effectiveness of LEAD. Each learning community will

1. integrate LEAD applications into their domain of activity,
2. develop new applications,
3. create performance outcomes for both education and research,
4. provide feedback to LEAD developers so that user needs drive the design and development process,
5. determine potential relevance of LEAD technologies to other related fields.

Furthermore, LEAD Learning Community members will serve as liaisons to the broader communities of education and research.

The two main LEAD Learning communities are the Teaching and Learning Community and the Research and Applications Learning Community. The Teaching and Learning LEAD Learning Community comprises of Pre-college educators, university faculty, undergraduate students, graduate students, DLESE member representatives, members of LEAD development, and members of the LEAD meteorology group. This community focuses on activities related to teaching and learning that are important to all levels of the education enterprise ranging from middle school to graduate school. These activities include but are not limited to the deployment and integration of LEAD applications as they are developed into the various learning environments, and to the evaluation and assessment of these applications. The assessment and evaluation process will be used to generate recommendations that will be communicated to the developers to drive further modification and refinement.

Research and Applications LEAD Learning Community comprises of representatives from the Development Testbed Center (DTC) at NCAR, NCEP, NOAA Forecast Systems Laboratory, etc. This community focuses on issues related to meteorology and computer science research and applications, particularly those requiring advanced cyberinfrastructure capabilities that otherwise are not available. This learning community will foster an interaction between the basic and applied research communities utilizing LEAD capabilities and LEAD developers.

2.2 Requirements for Education Users

Requirements for users in grades 6-12 (includes students and teachers):

1. Direct and easy access (Web-based) to real-time data and visualization tools
2. Direct and easy access to users guides, instructional materials, and modules
3. Capability to create just-in-time directories (be able to save specific information along the way in a personal directory for future use)
4. Capability to combine data and information for discovery (customize content)
5. Capability to compare model and observations
6. Remote access to large data stores and historical data

Some other functional requirements include:

1. Response time needs to be short
2. Allow multiple users at one time
3. Aesthetics issues and easy to navigate
4. Identify the types of systems and environments that the users would be using
5. Identify specific browser features that the users would need
6. Deliver content based on the user (so a 6-12 user may be able to access certain information in comparison to the other users. This may help with the ease of navigation).

2.3 Template Use Case Scenario

Use case for an Intermediate level user

1. User should be able to login to the portal with the user name and password provided, Portal should be able to retrieve the grid proxy from myproxy server (changes based on the new security model).
2. User should be able provide name and description of the experiment along with information related to sharing the results (publishing) to group, public or wishes to keep it private. Facility must be provided to change this information anytime.
3. Options should be presented to the user for selecting the experiment forecasting area by means of graphical tools (like geo-referencing GUI) or by manually entering the grid co-ordinates.
4. User should be able to perform the simulations for forecasting based on the latest available observational data or facility to be provided to research on forecasts for previous dates.
5. User should be able to enter the forecasting grid resolution.
6. Based on the forecasting area and date and time, portal should be able to query for observational data and transfer the files to the compute servers.
7. User should be presented with options of compute resources and options must be provided to change the execution servers for individual components.
8. Once the user configuration is complete the components should be able to execute on the selected compute resources.
9. User should be able to monitor the progress of the runs
10. Visualization options must be presented to the user to visualize the resultant output data.

3 Portal Framework Functional Requirements

3.1 Introduction

This section defines the requirements of the LEAD Portal "framework". By framework, we mean the entire stack of software required to host LEAD portlet applications. Thus, the "framework" includes not only the portal/portlet container, but also the web server and perhaps also the database backing store. However, most of these requirements apply only to the portal/portlet container.

3.2 General Requirements

The LEAD Portal framework must be based on open standards and must be able to be implemented using open source software (see section 3.3, *Design Principles of the LEAD Implementation Plan*).

3.3 User Management

The LEAD Portal framework must allow administrators to create, update and disable accounts. The framework should also allow users to create and manage their own accounts. Users should also be able to manage their group memberships.

3.4 Group Management

The LEAD Portal framework must support the concept of user groups. This will allow portal content creators to publish or push their content to a group or groups. For example, let's say we have an Administrators group; the framework should allow us to publish an account management portlet to that group's set of portlets. This functionality is important since different user groups have different needs and skill levels.

Lead members of a group should also be able to grant and revoke membership to their group members.

3.5 Security

The LEAD Portal framework must have all transactions conducted over a secure, encrypted channel (i.e., SSL over HTTP).

The framework must also support grid authentication mechanisms. That is, if a user is a member of the grid on which the LEAD Portal is hosted, the framework should allow for the user to authenticate to the portal via standard grid authentication mechanisms (e.g., using the supplied username and password to retrieve a proxy certificate from a MyProxy server). Alternatively, it is sufficient for the LEAD Portal framework if the user need only authenticate once to the LEAD Portal to be able to use any of the grid resources available from the portal.

The framework must keep user accounts private to each user. That is, user X should not be able to access, update, or disable the account of user Y as long as user X has taken proper precaution in keeping their password private, etc.

3.6 Customizability

The LEAD Portal framework must allow LEAD portal developers to customize the look and feel of the LEAD Portal, as designated in the section title "User Interface Design Requirements". The framework should also allow users to change the look and feel of their individual account as well as the layout and inclusion of extra portlets.

3.7 Inter-Portlet Communication

The LEAD Portal framework must allow for portlets to share data amongst themselves in the context of a user's portal session. That is, if a user performs an action in one portlet, that action can inform another portlet available to the user of some piece of data, and that other portlet can consequently alter its presentation or handling of actions processed accordingly.

4 Portlet Functional Requirements

4.1 Experiment Builder (MyExperiments)

The Experiment Builder portlet should have the following capabilities:

1. Experiment Summary page

- (a) Allow user to select an investigation in which to view experiments.
- (b) By default, show either the user's most recently worked with investigation, or the user's preferred investigation.
- (c) List all experiments within this investigation, with the experiments sorted in reverse by time of last status update.
- (d) The list of experiments should be chunked in groups, with "First", "Prev", "Next", and "Last" navigational links between the various chunks. There should be an indicator of how many chunks there are and how the current chunk relates with the larger list of experiments (e.g., "21-40 of 76").
- (e) For each experiment, the summary page should list the experiment's name, all and/or part of its description, the time of when the experiment's status was last updated, and the most recent status of the experiment.
- (f) Allow the user to open an experiment, which will take the user to the experiment view or edit page.
- (g) Allow the user to delete an experiment.
- (h) Allow the user to create a new experiment, which will take the user to the experiment edit page.
- (i) Allow the user to clone an existing experiment, which will take the user to the experiment edit page with several or all fields filled in with the values of the experiment being cloned.
- (j) *Allow the user to publish a completed experiment and its results* (Perhaps this would better be in the MyWorkspace portlet?)

2. Experiment Edit page

- (a) Allow the user to enter/update a name and description for the experiment.
- (b) Allow the user to enter notes about the experiment. Allow user to update previous notes, but keep change history.
- (c) Display the status and the time of last status update (which may be blank) for the experiment.

- (d) Allow the user to search for and manage the experiment's list of data sources.
- (e) Allow the user to search for and manage the experiment's list of data stores and other resources.
- (f) If the workflow has been selected, for each workflow parameter requiring binding, allow the user to search for data sources, data stores, or other resources using a tool that prefills the search criteria.
- (g) Allow the user to select a workflow, or create a new one. For each workflow, display its description.
- (h) When the user has selected a workflow, indicate what parameters need to be set for the workflow.
- (i) When the user selects a workflow, automatically bind the workflow parameters to data sources or data stores that the user has already selected.
- (j) Allow the user to cancel, save, or start an experiment. Cancelling means returning to the experiment summary page. Save means saving the partially completed experiment (to be completed at a later time) and returning to the experiment summary page. Starting the experiment means launching the workflow, and going to the experiment status page.
- (k) When the user launches a workflow, validate the workflow bindings and report errors to the user, indicating what must be updated before the workflow can be successfully launched.
- (l) Allow the user to search for similar experiments, i.e., experiments that have used the same workflow. Allow the user to see the results of such a search without having to leave the edit session.

3. Experiment Status page

- (a) Display the experiment's name, description, the experiment's time of most recent status update, and the most recent status.
- (b) Allow the user to enter notes about the experiment. Allow user to update previous notes, but keep change history.

- (c) Display all notifications that have been received. For each notification, display the notification's status, part or all of the message, and the timestamp of the notification.
- (d) Allow user to see more details on a notification, if available.
- (e) Update to show more notifications as they arrive.
- (f) Display notifications in reverse chronological order, chunking as described in [1d](#) above as necessary.
- (g) Display the experiment's workflow, it's description, it's parameters and their bindings.
- (h) Allow the user to get more information about the workflow, it's parameters and it's bindings (e.g., user selects a data source that is bound to a workflow parameter to find out where that data came from).
- (i) Allow the user to navigate back to the Experiment Summary page.
- (j) Allow the user to clone this experiment, taking the user to the Experiment Edit page with some or all fields filled in from this experiment.
- (k) *Allow the user to publish a completed experiment and its results* (Perhaps this would better be in the MyWorkspace portlet?)

4.2 Workflow Composer

The workflow composer should have the following capabilities:

1. Allow the user to retrieve a saved workflow template from MyLEAD (IP-UFR-Or6).
2. Allow the user to save workflow templates to MyLEAD (IP-UFR-Or5).
3. Allow the user to add and delete services from the workflow template palette. Allow the user to wire together the services (IP-UFR-Or6).
4. When wiring together services, indicate if there is an interface mismatch, that is, indicate if the output of one service is not appropriate as the input of another service. If there is a mismatch inform the user of the mismatch and allow the user to wire together service even though there is a mismatch. Visually, distinctively indicate a connection in the composer that is mismatched (IP-UFR-Or12).

5. Allow the user to create/update a name and description for a workflow template (IP-UFR-Or6).
6. When invoking the composer, it should automatically connect to a default MyLEAD server. However, the user can override this by configuring manually the MyLEAD server location.
7. Allow the user to configure default settings within each workflow component.
8. Allow the user to request a performance estimate for the workflow displayed in the palette (IP-UFR-Or14).
9. Allow the user to query for existing templates.
10. Allow the user to logically “group” together a subset of the workflow and publish the subset as a distinct workflow.
11. Allow the user to drop whole workflows onto the palette and wire together preexisting workflows (IP-UFR-Or7).
12. When wiring together whole workflows, allow the user to disassociate (“ungroup”) an added workflow and reconfigure it.

4.3 Workflow Browser

The workflow browser should have the following capabilities:

1. Show the status of the executing workflow. User should be able to tell at a glance which workflow components have finished, which are currently working, which have not started and which have failed (IP-UFR-Or8).
2. Allow user to select a component and see further details including the configuration for that component, any runtime configurations, notification messages sent by this component, log messages generated by this component, etc.
3. Allow a user to cancel a workflow or a subtree of a workflow (IP-UFR-Or8).
4. Allow the user to view checkpoints (IP-UFR-Or11).

5. Allow the user to request a checkpoint and optionally suspend the workflow (IP-UFR-Or11). Additionally, allow the user to restart a workflow from a selected checkpoint.
6. Allow the user to modify the configuration of workflow components of a running workflow (IP-UFR-Or11). This applies to workflow components that haven't started running yet.
7. Additionally, allow the user to modify the configuration of workflow components of a running workflow that are running or have already completed (IP-UFR-Or11).
8. Show the performance estimate for the workflow (IP-UFR-Or14). Show both aggregate and per component performance estimate information.
9. Allow the user to view the intermediate products of the workflow (IP-UFR-Or15). The browser should invoke the appropriate viewer for the results or redirect the user to a portlet within the portal that can visualize the intermediate results.
10. Allow the user to zoom in/out when viewing the workflow.

4.4 MyWorkspace (MyLEAD Browser)

The MyWorkspace Portlet should have the following capabilities:

1. Provide a view of the user's MyLEAD space including the user's workflow templates, investigations, experiments, collections, etc.
2. Provide a means of querying the user's MyLEAD space.
3. Allow the user to view and query over the public MyLEAD spaces of other users (IP-UFR-D6).
4. Allow the user to make public some selected subset of his or her MyLEAD space (IP-UFR-D6).
5. Allow the user to specify exactly to which users or which group of users a subset of his or her MyLEAD space is published (IP-UFR-D6).
6. Allow the user to specify the duration of access to a published subset of the user's information space (IP-UFR-D6).

7. Allow the user to download items from the user's personal information space to his or her workstation.
8. Allow the user to upload items from the user's workstation to his or her MyLEAD space.
9. Allow the user to set preferences of "favorites". Some examples includes setting a preferred archival data storage system, preferred visualization tool, etc.

4.5 Registry Catalog

The Registry Catalog Portlet should have the following capabilities:

1. Allow the user to set up a preferred registry catalog.
2. Allow the user to query for data sources based on the following criteria:
 - Title
 - Subject
 - Description
 - Creator
 - Publisher
 - Data Format
 - Access Protocol
3. In addition, allow the user to query over all fields with one search
4. In addition, allow the user to specify a temporal range for the query
5. In addition, allow the user to specify a spatial range for the query, using a graphical tool
6. Allow the user to search for compute and storage resources based on the following criteria:
 - Name
 - Protocol

- Storage space
 - Type of resource
 - Computational power
7. When returning the results of a query, any results within the user's preferred data sources or compute and data storage resources should be returned first and visually distinguished from the other results.

4.6 Resource Management (User Preferences)

(I think this is actually part of MyWorkspace or perhaps Account Management).

4.7 Account Management

The Account Management Portlet should have the following capabilities:

1. Allow the user to update the account password.
2. Allow the user to update any personal information stored for the account.
3. Allow the user to add other users under this account.
4. Allow the user to set global preferences.
5. Allow the user to update his or her primary group membership (classification). For example, allow a user to change her primary group membership from "Students" to "Teachers".

There is some difficulty here in that the portal framework may provide its own account management interface with some overlapping functionality to what is described here. In that case, this section describes what account management capabilities must be present in the LEAD portal, whether or not they exist in the same portlet.

4.8 Application Code Management

For advanced users, revision control of code used within applications, i.e., used by workflow components.

4.9 Other Tools

4.9.1 Visualization Services

The LEAD portal should provide users with a choice of visualization options. These visualization services come in two flavors: those that are rendered on server-side resources and then displayed within the portal (e.g., NCL), and those that are rendered on client-side resources and displayed outside the portal on the user's workstation (e.g., IDV). The following capabilities need to be present in the LEAD portal:

1. Provide the user with a server-side rendering option that then displays the visualization using a GIF, JPEG, or PNG (or some other broadly supported image format) image in the portal. (This is to accommodate user's who have modest workstation resources for whom we can assume only that they have a recent standards-compliant web browser.)
2. Provide the user with a client-side rendering option that starts up a desktop tool and directs it to the data to be visualized.
3. Allow the user to configure the portal to invoke a new desktop tool for doing visualization work.
4. Allow the user to visualize not only end products but also intermediate workflow results (IP-UFR-Or15).

4.10 Not Touched On

The following is a list of functional requirements for portlet functionality that haven't been dealt with in this section.

1. Launch a workflow based upon trigger conditions (IP-UFR-Or10). There should be some sort of portlet interface to allow users to create, view, update, disable, enable, and delete workflow triggers, but it is not clear now how that might manifest.

5 User Interface Design Requirements

5.1 Personalization

5.1.1 Overall Look and Feel

1. The default Portal should follow the overall LEAD standard colors, blue and white and should display the LEAD and NSF logos on the login page.
2. The user should have the flexibility to change the colors, fonts, content, icons and size of the various portlets displayed within the portal window. (The extent of this flexibility will be a direct factor of the portal framework chosen which will be discussed in the portal implementation plan.)
3. A live weather feed should be displayed on the portal.
4. Links to real-time and historical severe weather forecasts should be provided on the login page.
5. Severe weather alerts should be presented to the user in a noticeable way.

5.1.2 Flexibility in Design

The content of the portal and the embedded portlets discussed in chapter 3 should be portable enough to be deployed in a new framework with a fairly minimum effort. The design and development of the various components of the LEAD-Portal should be made so that they are exportable to different other interfaces if needed.

5.1.3 Portal Interface and User Groups

- The Portal Interface should be presented to the user based on their group. Each group content should be different based on the user groups knowledge and expertise determined by the LEAD user policy.
- If the user requests for a change of group and once approved by the portal administrator, user should be presented with user interface pertaining to the new group but should retain all experiments and data

performed earlier. Eg. I gained enough experience and wishes to perform experiments in more advanced level but would like to compare the results with my current experiments.

5.2 Minimum Computer/Browser Requirements

1. The computer resource requirements by a user should be as minimal as possible.
2. Essential requirement should be connectivity to Internet and a web-browser.
3. Most of the available stable browsers should be supported (browsers which can support HTML, Javascript and which have 128 bit SSL encryption for https).
4. Portal should be functional from Windows, Linux, Unix and Mac desktops and laptops.
5. Reliance on Plug-ins and client software like webstart should be kept at a very minimal level and must be avoided if possible, where it would prevent a user not having the necessary software installed from accessing a LEAD subsystem. However users are recommended to use the latest stable versions of browsers and install the latest Java Virtual Machine (JVM) on their clients to access the maximum functionality everything offered by each subsystem through the portal.