



# MyLead Release V1.2 Developer's Guide (Client Service)

Project Title: myLEAD

Document Title: myLEAD Release V.1.2 Developer's Guide

Organization: Indiana University

Date: March, 24, 2006

Contact: Sangmi Lee Pallicaka ([leesangm@cs.indiana.edu](mailto:leesangm@cs.indiana.edu))

Authorship: Beth Plale, Sangmi Lee Pallickara, Scott Jensen, and Yiming Sun

---

1	Introduction .....	2
1.1	myLEAD License .....	2
2	System Specification .....	3
2.1	Prerequisite Products .....	3
2.2	Required Libraries .....	3
3	Access to the MyLead Agent Service .....	4
3.1	Generating and Parsing Message .....	4
4	MyLead Agent APIs .....	4
4.1	Experiment APIs .....	4
4.1.1	Creating a Project .....	5
4.1.2	Creating an Experiment .....	5
4.1.3	Creating a Collection .....	5
4.1.4	Register a File.....	6
4.2	Administrative APIs .....	6
5	Sample Code .....	7

## 1 Introduction

This document provides a guide to the developers accessing mylead server through the mylead client service. The document covers:

- How to setup the client environment
- How to access the mylead agent service
- Description of mylead agent APIs
- 

You can start with downloading myleadStarterKit1.2.tar from distribution site. This sample code is also available in mylead client service package (myleadClientService1.2.tar).

### 1.1 myLEAD License

The file doc/myLEAD-Licence.txt within the source distribution directory contains the product license. Make sure that you read this license and accept its conditions before continuing.

## 2 System Specification

myLEAD is designed to work on – and has been tested under following platforms:

- Redhat Linux Gentoo

### 2.1 Prerequisite Products

Before you start develop a requester to mylead agent service, the mylead system should be installed and running properly. The mylead system consists,

- JDK 1.4X
- ANT 1.6X
- WS Notification/Eventing Broker

WS-Messenger(WSMG) is WS-Notification & WS-Eventing implementation.  
See <http://www.extreme.indiana.edu/xgws/messenger/index.html>

About the installation of the mylead system, please refer the installation guide.

### 2.2 Required Libraries

Note: You can find all of the required libraries in the myleadClientService1.2.tar file.  
Please download from mylead release web site.

JAR Name	Used For	Package Version	URL
xmlbeans-1.0.jar	Process XML message	1.0	<a href="http://xmlbeans.apache.org/">http://xmlbeans.apache.org/</a>
xpp3-1.1.3.4.N.jar	XML parser	3-1.1.3.4.N	<a href="http://www.extreme.indiana.edu/xgws/xsoap/xpp/mxp1/index.html">http://www.extreme.indiana.edu/xgws/xsoap/xpp/mxp1/index.html</a>
MyleadCSinterface-1.2.jar	MyLead agent interface	1.2	In mylead distribution site
GSX-1.2_SC04_20041102.jar	Grid toolkit	1.2	<a href="http://www.extreme.indiana.edu/xgws/GSX/">http://www.extreme.indiana.edu/xgws/GSX/</a>

### 3 Access to the MyLead Agent Service

First, please make sure you have valid account in mylead server. Also make sure that mylead server, agent service, RLS server and notification(WS-Eventing brokering service) server are accessible. To test your setup, mylead installation test suite is available in the distribution site. Please download myleadInstallationTest0.3alpha.tar and check your environment. Installation test suite is also available in the agent service package. For more information, please refer the installation guide.

To access mylead agent service, service requesters should include,

```
import edu.indiana.extreme.gsx.util.ServiceInvoker;
import edu.indiana.extreme.lead.types.*;
import edu.indiana.dde.mylead.agent.MyLeadAgentInterface;
```

The mylead agent service is invoked by ServiceInvoker. For example,

```
MyLeadAgentInterface myleadAgentService =
    (MyLeadAgentInterface)ServiceInvoker.
    createXBeansRpcProxy(service_url,MyLeadAgentInterface.class);
```

#### 3.1 Generating and Parsing Message

MyLead agent service follows data schema used in All-hands-meeting in May, 12, 2005. Each XML schema is generated and parsed by Java APIs generated by XMLBeans. Class files, source files, and schema are available at download page. Please note that this schema may change after the LEAD schema is published.

MyLead client package includes sample code, `ClientTestMessageGenerator.java`.

### 4 MyLead Agent APIs

The myLEAD v 0.3 alpha provides,

- Experiment APIs
- Administrative APIs

#### 4.1 Experiment APIs

MyLead agent service maintains the finite state machine to monitor and manage activities related to creating new collections and registering files. The requests are processed in the sequence of,

```
Creating an active experiment
→ Setup the active experiment
→ Create collection and register files
→ Workflow finished or close/shutdown the active experiment
```

Above sequence is not exactly same with the finite state machine in the mylead agent service. They are simplified steps for developers. For more information about the finite state machine, please refer our publication<sup>1</sup>.

### 4.1.1 Creating a Project

```
public String createProject(String uID, String leadResource, boolean assignNewResourceID)
```

Project is the highest element in the user's personal workspace. MyLEAD provides flexible organization to the users and services. However, there are very simple requirements to build a structure, such as project and experiment. To create valid project space, user should provide a valid user id, lead metadata for the project. The lead metadata is passed to the mylead client service as a String object. It MUST be a valid lead metadata, otherwise, mylead server will not process the data and throw an exception. MyLEAD client service assigns new resource ID to the leadResource if there is resourceID i.e. unique GUID in the leadResource, if the parameter assignNewResourceID is set as true. If the creation of the project is successful, mylead returns the resourceID of the new object as a String. This resourceID can be used to create child elements of this project.

Project can contain collection, experiment, and files. Users or services can combine those elements to build their own structure.

### 4.1.2 Creating an Experiment

```
public String createExperiment(String uID, String leadResource, String parentProjectResourceID,
                             boolean assignNewResourceID)
```

The experiment is the biggest container for each experiment in mylead. Requesters must provide valid userID, metadata and reference of it's parent. The reference of the parent is specified as the resource ID of the parent. There can be multiple experiments under a project.

### 4.1.3 Creating a Collection

```
public void createCollection(String uID, String leadResource, String parentProjectResourceID,
                            boolean assignNewResourceID)
```

---

<sup>1</sup> Sangmi Lee Pallickara, Beth Plale, Scott Jensen, Yiming Sun, Monitoring Access to Stateful Resources in Grid Environments, *To appear IEEE International Conference on Services Computing*, Orlando, Florida, July 2005

Collection provides flexible structure in mylead personal metadata catalog. Users can build a collection under project, experiment, and even collection. Like experiment, users or services **MUST** provide valid user ID, metadata and the parent's resourceID. Collections can contain nested collections and files as its child. We will show how to register a file in the next section.

#### 4.1.4 Register a File

```
public void registerFile(String uID, String leadResource, String dtatsourcelocation,
                        String parentProjectResourceID, boolean assignNewResourceID)
```

Registering a file in mylead contains three steps,

- Register to the name resolving service, such as RLS
- Transfer the file from user's location to the mass storage repository
- Register the metadata of file to mylead service

First and second steps are processed by access to the 3<sup>rd</sup> party services, such as TDR or DRS. After the file is transferred successfully and the name resolver contains valid information about the file, mylead will register the metadata to the mylead server. Users or service **MUST** provide valid user id, metadata, location where the data is resigned, and parent's resource id. In this version, we ignore the first and second steps. As soon as those services are ready to use, we are planning to provide full functionality.

## 4.2 Administrative APIs

MyLead v0.3 alpha provides a set of APIs for mylead administrator.

```
public MyLeadServiceInfoArrayDocument getMyLeadServiceInfo()
```

This method provides the information about mylead distributed services. MyLead system will be deployed as a master-satellite like distributed service in the future release. This API provides information about all MyLead systems over the MyLead network.

```
public void setMyLeadServiceInfo(MyLeadServiceInfoDocument myleadsrvinfo)
```

This method modifies the information of mylead system which already exists.

```
public void addMyLeadServiceInfo(MyLeadServiceInfoDocument myleadsrvinfo)
```

This method creates a mylead system which is newly joined in the distributed mylead system.

```
public MyLeadUserInfoArrayDocument getAllUserInfo()
```

This method returns user information of all of distribute mylead system.

```
public MyLeadUserInfoDocument getUserInfo(String userid)
```

This method queries information about single user with `userid`.

```
public void setUserInfo(MyLeadUserInfoDocument userinfo)
```

This method modifies a user's information which already exists in the mylead system.

```
public void addUserInfo(MyLeadUserInfoDocument userinfo)
```

This method creates a new user in the mylead system.

```
public StringArrayDocument getActiveExperiment(String uID, String proj)
```

This method returns all of the active experiments running in the mylead system.

```
public StringArrayDocument getActiveProject(String uID)
```

This method returns all of the active projects running in the mylead system.

## 5 Sample Code

ClientTest.java : `src/java/sample/ClientTest.java`

Note: You can find this sample code in the `myleadClientService1.2.tar` or `myleadStartersKit1.2.tar` from the myLEAD release web page.

```
import edu.indiana.extreme.gsx.util.ServiceInvoker;
import edu.indiana.extreme.lead.types.*;
import edu.indiana.dde.mylead.agent.MyLeadAgentInterface;

public class ClientTest {

    private MyLeadAgentInterface myleadAgentService;

    public ClientTest(String service_url) throws Exception{
        initServiceClient(service_url);
    }

    /**
     * initializing the mylead client service
     */
    public void initServiceClient(String service_url) throws Exception{
        myleadAgentService =
            (MyLeadAgentInterface)ServiceInvoker.
                createXBeansRpcProxy(service_url, MyLeadAgentInterface.class);
    }
}
```

```
/**
 * a method to create a project in mylead.
 *
 * user has to provide a valid userID, String which is a valid leadmetadata
 * describing the project. If the user puts true in the newGUID parameter,
 * mylead client service will assign a new resource ID for this project.
 */

public String addProject(String uid,
                        String metadata,
                        boolean newRscID){

    try{
        String rscID = myleadAgentService.createProject(uid,metadata,newRscID);
        System.out.println(rscID+" is created successfully.");
        return rscID;
    }catch(Exception e){
        System.out.println(e);
    }
    return null;
}

/**
 * a method to create an Experiment in mylead.
 *
 * user has to provide a valid userID, String which is a valid leadmetadata
 * describing the experiment. When the user adds an experiment, user should
 * specify the parent of this experiment. The parent of this experiment is
 * simply specified by the resourceID of the parent.
 * If the user puts true in the newGUID parameter,
 * mylead client service will assign a new resource ID for this experiment.
 */

public String addExperiment(String uid,
                            String metadata,
                            String parent,
                            boolean newRscID){

    try{
        String rscID = myleadAgentService.createExperiment(uid,metadata,parent,newRscID);
        System.out.println(rscID+" is created successfully.");
        return rscID;
    }catch(Exception e){
        System.out.println(e);
    }
    return null;
}

/**
 * a method to create a collection in mylead.
 *
 * user has to provide a valid userID, String which is a valid leadmetadata
 * describing the collection. When the user adds a collection, user should
 * specify the parent of this collection. The parent of this collection is
 * simply specified by the resourceID of the parent.
 * If the user puts true in the newGUID parameter,
 * mylead client service will assign a new resource ID for this collection.
 */
```

```
public String addCollection(String uid,
                          String metadata,
                          String parent,
                          boolean newRscID){
    try{
        String rscID = myleadAgentService.createCollection(uid,metadata,parent,newRscID);
        System.out.println(rscID+" is created successfully.");
        return rscID;
    }catch(Exception e){
        System.out.println(e);
    }
    return null;
}

/**
 * a method to add a file.
 *
 * user has to provide a valid userID, String which is a valid leadmetadata
 * describing the collection. If the user wants to add a file to mylead
 * actual file is also stored in the mass storage repository. myLEAD client service
 * accesses DRS service which pulls the file from the user's location to the
 * mass stroage service. Therefore, the user should specify the location of
 * the actual file to the mylead client service.
 * Also, user should specify the parent of this file. The parent of this file is
 * simply specified by the resourceID of the parent. If you put "" for parent parameter
 * your file will be stored WHITEBOARD space of your workspace.
 * If the user puts true in the newGUID parameter,
 * mylead client service will assign a new resource ID for this file.
 *
 * NOTE: for this version, mylead does not transfer actual file. This functionality
 * will be available after file transfer mechanism is ready to use. (DRS and/or TDR)
 */

public String addFile(String uid,
                    String metadata,
                    String datasourcelocation,
                    String parent,
                    boolean newRscID){
    try{
        String rscID =
myleadAgentService.registerFile(uid,metadata,datasourcelocation,parent,newRscID);
        System.out.println(rscID+" is added successfully.");
        return rscID;
    }catch(Exception e){
        System.out.println(e);
    }
    return null;
}

public static void main(String args[]) throws Exception {

    // PLEASE MAKE SURE YOU HAVE AN ACCOUNT IN MYLEAD SERVER
    String uid = "leesangm"; //please replace your valid user DN

    // PLEASE MAKE SURE MYLEAD AGENT SERVICE IS RUNNING
```

```

String serviceURL = "http://bitternut.cs.indiana.edu:10085/mylead";

/* These calls create project with the metadata below and create an example under the
 * project with same metadata. And a collection is added to the experiment and a file
 * is added to the collection. All of the resourceID is generated by the mylead
client
 * service.
 */

try{
    // BE READY TO ACCESS MYLEAD AGENT.
    ClientTest ct = new ClientTest(serviceURL);
    String metadata = ClientTest.leadObjectPt1+"1000"+ClientTest.leadObjectPt2;

    String receivedRscID = ct.addProject(uid, metadata, true);
    receivedRscID = ct.addExperiment(uid, metadata,receivedRscID, true);
    receivedRscID= ct.addCollection(uid, metadata, receivedRscID, true);
    receivedRscID= ct.addFile(uid, metadata,
"http://www.cs.indiana.edu/~leesangm/test.txt",
        receivedRscID, true);
    }catch (Exception e){
        System.out.println(e);
    }
}

/**
 * Metadata below is an example to help using MyLEAD client service APIs.
 * Please note that the
 * name and contents are not real data.
 */

static String leadObjectPt1 = "<lead:LEADresource xmlns:lead=\"LEAD\"
xmlns:elem=\"LEADElements\" " +
    " xmlns:fgdc=\"FGDC\" > \n" +
    " <elem:resourceID xmlns:lead=\"LEADElements\">";

static String leadObjectPt2 = "</elem:resourceID> \n" +
    " <lead:data> \n" +
    " <lead:idinfo> \n" +
    " <lead:citation> \n" +
    " <fgdc:origin>AAA</fgdc:origin> \n" +
    " <fgdc:origin>BBB</fgdc:origin> \n" +
    " <fgdc:pubdate>Unknown</fgdc:pubdate> \n" +
    " <fgdc:title>CCCCCCCC</fgdc:title> \n" +
    " <fgdc:pubinfo> \n" +
    " <fgdc:pubplace>DDDDDDD, EE</fgdc:pubplace> \n" +
    " <fgdc:publish>FFFFFFFFF</fgdc:publish> \n" +
    " </fgdc:pubinfo> \n" +
    " <fgdc:othercit>GGGGGGGGGGGG</fgdc:othercit> \n" +
    " </lead:citation> \n" +
    " <lead:descript> \n" +
    " <fgdc:abstract>HHHHHHHHHHHH</fgdc:abstract> \n" +
    " <fgdc:purpose>IIIIIIIIIIII</fgdc:purpose> \n" +
    " </lead:descript> \n" +
    " <lead:status> \n" +
    " <fgdc:progress>In work</fgdc:progress> \n" +

```



```
"      </fgdc:cntperp> \n" +
"      <fgdc:cntpos>Lawyer</fgdc:cntpos> \n" +
"      <fgdc:cntaddr> \n" +
"          <fgdc:addrtype>mailing</fgdc:addrtype> \n" +
"          <fgdc:address>Any University</fgdc:address> \n" +
"          <fgdc:address>101 Any Street</fgdc:address> \n" +
"          <fgdc:city>Some City </fgdc:city> \n" +
"          <fgdc:state>My State</fgdc:state> \n" +
"          <fgdc:postal>zipitty do da </fgdc:postal> \n" +
"          <fgdc:country>USA</fgdc:country> \n" +
"      </fgdc:cntaddr> \n" +
"      <fgdc:cntaddr> \n" +
"          <fgdc:addrtype>physical</fgdc:addrtype> \n" +
"          <fgdc:address>School of Hard Knocks</fgdc:address> \n" +
"          <fgdc:address>100 Blues Avenue</fgdc:address> \n" +
"          <fgdc:city>Memphis</fgdc:city> \n" +
"          <fgdc:state>TN</fgdc:state> \n" +
"          <fgdc:postal>11111</fgdc:postal> \n" +
"          <fgdc:country>USA</fgdc:country> \n" +
"      </fgdc:cntaddr> \n" +
"      <fgdc:cntvoice>123-456-7890</fgdc:cntvoice> \n" +
"      <fgdc:cntvoice>111-222-3333 ext.7890</fgdc:cntvoice> \n" +
"      <fgdc:cnttdd>777-888-9999</fgdc:cnttdd> \n" +
"      <fgdc:cntfax>321-654-0987</fgdc:cntfax> \n" +
"      <fgdc:cntfax>321-654-1122</fgdc:cntfax> \n" +
"      <fgdc:cntemail>johndoe@shmoe.edu</fgdc:cntemail> \n" +
"      <fgdc:cntemail>johnny@shmoeco.com</fgdc:cntemail> \n" +
"      <fgdc:hours>After midnight</fgdc:hours> \n" +
"      <fgdc:cntinst>knock 3 times on the ceiling</fgdc:cntinst> \n" +
"  </fgdc:cntinfo> \n" +
" </fgdc:distrib> \n" +
" <fgdc:stdorder> \n" +
"   <fgdc:digform> \n" +
"     <fgdc:digtinfo> \n" +
"       <fgdc:formname>netCDF</fgdc:formname> \n" +
"       <fgdc:formvern>2.50</fgdc:formvern> \n" +
"       <fgdc:formspec>Some netCDF Spec</fgdc:formspec> \n" +
"       <fgdc:formcont>Some netCDF Format Content</fgdc:formcont> \n" +
"       <fgdc:filedec>No compression applied</fgdc:filedec> \n" +
"       <fgdc:transize>35.2</fgdc:transize> \n" +
"     </fgdc:digtinfo> \n" +
"   </fgdc:digform> \n" +
" </fgdc:stdorder> \n" +
" <fgdc:stdorder> \n" +
"   <fgdc:digform> \n" +
"     <fgdc:digtinfo> \n" +
"       <fgdc:formname>HDF</fgdc:formname> \n" +
"       <fgdc:formvern>3.50</fgdc:formvern> \n" +
"       <fgdc:formspec>Some HDF Spec</fgdc:formspec> \n" +
"       <fgdc:filedec>Compressed so tight it hurts</fgdc:filedec> \n" +
"     </fgdc:digtinfo> \n" +
"   </fgdc:digform> \n" +
" </fgdc:stdorder> \n" +
" </lead:distinfo> \n" +
" <lead:dataqual> \n" +
"   <fgdc:complete>Yea, there is some stuff missing</fgdc:complete> \n" +
"   <fgdc:lineage> \n" +
```

```
"      <elem:procstep> \n" +
"      <elem:procdesc>The first of many steps</elem:procdesc> \n" +
"      <elem:procDateTime>2005-12-05T11:12:13</elem:procDateTime> \n" +
"      <elem:srcused> \n" +
"          <elem:resourceID>idd.model.eta_201</elem:resourceID> \n" +
"      </elem:srcused> \n" +
"      <elem:srcused> \n" +
"          <elem:resourceID>idd.model.eta_202</elem:resourceID> \n" +
"      </elem:srcused> \n" +
"      <elem:srcprod> \n" +
"          <elem:resourceID>WRF_NCSA_100</elem:resourceID> \n" +
"      </elem:srcprod> \n" +
"      <elem:srcprod> \n" +
"          <elem:resourceID>WRF_NCSA_101</elem:resourceID> \n" +
"      </elem:srcprod> \n" +
"  </elem:procstep> \n" +
"  <elem:procstep> \n" +
"      <elem:procdesc>The second of many more steps</elem:procdesc> \n" +
"      <elem:procDateTime>2005-12-06T12:13:14</elem:procDateTime> \n" +
"      <elem:srcprod> \n" +
"          <elem:resourceID>idd.model.wrf</elem:resourceID> \n" +
"      </elem:srcprod> \n" +
"  </elem:procstep> \n" +
" </fgdc:lineage> \n" +
" </lead:dataqual> \n" +
" <lead:enclosedresources> \n" +
"   <elem:resourceID>idd.model.eta_201</elem:resourceID> \n" +
"   <elem:resourceID>idd.model.eta_202</elem:resourceID> \n" +
"   <elem:resourceID>WRF_NCSA_100</elem:resourceID> \n" +
"   <elem:resourceID>WRF_NCSA_101</elem:resourceID> \n" +
" </lead:enclosedresources> \n" +
" </lead:data> \n" +
"</lead:LEADresource>;
```

```
}
```