



ILS Installation Guide

Project Title: PGPGrid
Document Title: ILS Installation Guide
Document Identifier: EPCC-IS-PGPGRID-IG-ILS-1.1
Distribution Classification: Commercial in Confidence

Authorship: Paul J Graham
Approval List: PGPGrid team
Distribution List: PGPGrid team

Document History:

Personnel	Date	Summary	Version
Paul J Graham	02/09/2004	New release, incorporating changes to benchmark value definition	1.1
Paul J Graham, Gavin Pringle, Kostas Kavousannakis	04/08/2004	Release, incorporating comments from KK and GJP	1.0
Paul J Graham	14/07/2004	First Draft	0.1

1	Overview	2
1.1	Release Notes	2
1.2	Licence	2
2	Introduction	2
2.1	ILS Architecture	2
2.2	Limitations of this release	2
3	Requirements.....	2
3.1	Assumed User Background and Knowledge	2
3.2	System Specification	2
3.3	Prerequisites	2
3.3.1	All Components.....	2
3.3.2	Locator Only.....	2
4	ILS Installation	2
4.1	CLASSPATH configuration	2
4.2	Locator.....	2
4.2.1	Deployment	2
4.2.2	Configuration.....	2
4.2.3	Testing the web services.....	2
4.3	Servent.....	2
4.3.1	Servent Configuration	2
4.3.2	Servent Command Line Clients.....	2
4.4	Initiator	2
4.4.1	Initiator Resource Request	2
4.4.2	Initiator Command Line Client.....	2
5	Conclusions	2
6	References	2

1 Overview

This is the user guide for the installation and usage of the Initiator-Locator-Servent (ILS) resource location software, as part of the PGPGrid project workpackages 4 and 5[1].

1.1 Release Notes

This document refers to release 1.2 of the software. The previous release was 1.1.

Release	Notes
1.2	Correction to the comparison method for the benchmark value in the Locator (higher benchmark value is now interpreted as being lower performance)
1.1	Small changes to the exception handling and improved documentation in the code for the test cases
1.0	Initial release

1.2 Licence

See the accompanying document “PGPGridOpenSourceLicence.txt” included in the release.

2 Introduction

The motivation and requirements for the ILS software are described in more detail in [2] and [3]. In brief, the software consists of three components: the Initiator, the Locator, and the so-called “Servent”.

The Locator maintains a list of resources, and parameters associated with those resources, in a database. This list can be accessed and updated via web services [4], enabling the Locator to undertake activities such as resource registration, the acceptance of status updates for the resource, and to serve requests for resources that match a given set of job requirements. There are two web services, the Locator Registry and the Locator Request services. A Servent talks to the Registry, while an Initiator talks to the Request service.

The Servent (*Server Entity*) runs on a resource that is available for the execution of jobs. To this end it provides a mechanism for documenting the resource parameters (such as number of processors, memory etc), and communicating those parameters to a Locator, including dynamic information (such as busy/available for example). A list of the resource parameters can be found in [3] and is also included here in appendix A.2 in an example Servent configuration file.

The Initiator is used to request resources capable of and available for running jobs. It provides a mechanism for setting the job parameters (closely linked to the resource parameters) and sending these to the Locator, which in turn returns resources suitable for running the job. A list of the job parameters can be found in [3] and is also included here in appendix A.3 in an example resource request file.

It is intended that, once the Initiator has received a list of resources available for its jobs from the Locator, it will then contact and negotiate with those resources the actual running of the jobs and transfer of data etc via the Servent at each resource. However this functionality has not yet been implemented (see section 2.2).

2.1 ILS Architecture

The ILS interactions are based on Java Web Services utilising the Axis [5] messaging tools. Efforts have been made, however, to keep the ‘business logic’ of the components as separate as possible from the underlying communication mechanism. This should make it straightforward to incorporate any future changes or enhancements to the web services communications, or, indeed, to utilise other communication protocols. Further details of the architecture can be found in the high-level design document [6] and the User Guide [7], however, the essential knowledge to get started is detailed in sections 3 and 4 below.

2.2 Limitations of this release

It should be noted that while every care has been taken in the development of this software, it has not been comprehensively tested across a wide range of platforms or with different versions of the prerequisite tools (see section 3.3). As such it may not work with configurations that veer from those recommended.

In terms of functionality, this first release of the software is intended as a proof of concept. For example, currently it is left for the user to determine how to pass their job to the Servent resource and run it once a suitable resource has been identified. However, it is intended that this functionality may be included in future releases.

3 Requirements

In this section we identify the prerequisites required for using the ILS software.

3.1 Assumed User Background and Knowledge

This guide should make the use of the ILS straightforward, but it will be easier if the user is familiar with the following:

- Java programming
- Software installation under Unix or Windows
- Basic understanding of XML
- MySQL relational database
- Basic concepts of Web Services

3.2 System Specification

The software has been successfully tested under Microsoft Windows 2000 and Sun Solaris 9. In terms of hardware the software is not particularly computationally or bandwidth demanding as it deals mainly with the communication of small packets of information. However, one could envisage that the Locator would start making more significant demands for compute resource for larger databases (1000+ resources) and/or large volume of job requests (1000+/hour). This should be considered when choosing the Locator host machine in such scenarios.

3.3 Prerequisites

Here we list the prerequisite third-party products that must be installed and configured before the installation of the ILS software. Note that it is expected that the software will work correctly with later versions of the products than those listed here, but the versions here are those that have been tested against.

3.3.1 All Components

These are the products required by all three ILS components.

3.3.1.1 Java

All machines using the software require Java 1.4.2 or higher, available from <http://java.sun.com/>.

3.3.1.2 JUnit (optional)

There are suites of test cases included with the software that utilise the JUnit unit testing libraries. Whilst for general running of the software these are not required, they need to be installed if the developer wishes to use the unit tests. JUnit 3.8.1 or higher is available from <http://www.junit.org/>.

3.3.1.3 Axis

Apache Axis is required for the messaging between components. The Axis Jar files need to be added to the CLASSPATH for the java command (see section 4.1 below). They are downloadable from <http://ws.apache.org/axis/releases.html> (version 1.1).

3.3.2 Locator Only

These products are only required for a machine which is intended to be used as a Locator host.

3.3.2.1 Tomcat

The machine intended to act as the Locator requires Jakarta Tomcat 4.1.27 or higher to be installed for the web services hosting, available from <http://jakarta.apache.org/tomcat>.

3.3.2.2 Database

The software has been written to utilise any JDBC enabled database, however, it has been tested against, and the examples refer to, the relational database MySQL, available from <http://www.mysql.com/downloads>. This needs to be installed on a machine accessible by the Locator.

In addition to the database management system itself, a JDBC driver for the database being used needs to be visible to the web services. This JAR file needs to be placed where it is visible to the web services, as described below in section 4.2.1. The *Connector/J* driver available from <http://dev.mysql.com/downloads/connector/> is suitable for MySQL.

4 ILS Installation

This section describes the installation processes in turn for the Locator, Servent and Initiator components. In the following subsections, we assume the third-party products are installed with default settings. Further, we assume that the ILS release files have been downloaded and unzipped, with their location referred to as <ILS_HOME> in this document.

We introduce some of the example codes, which have been included with the distribution, for testing and demonstrating the ILS functionality. For details on how to incorporate the ILS software into your own code please see the User Guide [7].

Note that unless explicitly mentioned we use the “\” MS-DOS file separator in the examples below, which should be replaced with “/” under UNIX.

4.1 CLASSPATH configuration

For the correct functioning of the operations described below the Java CLASSPATH environment variable must be configured to include the Axis (section 3.3.1.3) and ILS (<ILS_HOME>\pgpgrid_ils.jar) JAR files. Environment variables can be set in the following ways: for UNIX (non-csh), we use the `export` command, e.g.:

```
export AXIS_LIB=$HOME/axis-1_1/lib
export CLASSPATH=$CLASSPATH:$AXIS_LIB/axis-ant.jar
export CLASSPATH=$CLASSPATH:$AXIS_LIB/axis.jar
...
export CLASSPATH=$CLASSPATH:$HOME/ILS/pgpgrid_ils.jar
```

Similarly we use the `setenv` command for UNIX (csh). For Windows we use the `set` command:

```
set AXIS_LIB=c:\axis-1_1\lib
set CLASSPATH=%CLASSPATH%;%AXIS_LIB%\axis-ant.jar
set CLASSPATH=%CLASSPATH%;%AXIS_LIB%\axis.jar
...
set CLASSPATH=%CLASSPATH%;c:\ILS\pgpgrid_ils.jar
```

To save retyping it may be worth adding these commands to your start up scripts under UNIX, or on Windows go to *Start->Settings->Control Panel->System->Advanced* to set the environment variables permanently.

4.2 Locator

The Locator consists of web services and a database and is, therefore, more complex to install than the other components. First of all, if not already there, Tomcat should be installed according to its instructions. Then we need a database. We assume MySQL but any JDBC enabled database should work, along with the appropriate JDBC drivers. Next install Axis as per its instructions. Once you are satisfied the products have been installed correctly and the CLASSPATH has been set (see section 4.1), it is time to deploy the Locator services.

4.2.1 Deployment

First the relevant JAR files need to be visible to the Tomcat server. Place the following files in to the directory where you placed the Axis libraries, for example in <TOMCAT_HOME>\webapps\axis\WEB-INF\lib where <TOMCAT_HOME> is the directory of the Tomcat installation.

```
<ILS_HOME>\pgpgrid_ils.jar
and
yourJDBCdriver.jar
(for example, for MySQL the driver suggested is mysql-connector-java-
3.0.12-production-bin.jar, or similar)
```

Restart your Tomcat server after this has been done.

To deploy the web services we use the Axis `AdminClient` tool. This requires that the Axis libraries be visible in the CLASSPATH, as discussed in section 4.1. Now type the following

in directory `<ILS_HOME>\Locator\Registry\` to deploy the Locator Registry web service:

```
java org.apache.axis.client.AdminClient
-lhttp://localhost:8080/axis/services/ILSRegistry
deploy.wsdd
```

And similarly type the following in directory `<ILS_HOME>\Locator\Request\` to deploy the Locator Request web service:

```
java org.apache.axis.client.AdminClient
-lhttp://localhost:8080/axis/services/ILSRequest
deploy.wsdd
```

For both cases replace “localhost” and “8080” with your hostname and Tomcat server port number as necessary.

The web services are now deployed and should be visible via a web browser at:

```
http://<YOUR_HOSTNAME>:<YOUR_PORT>/axis/services
```

4.2.2 Configuration

Before the web services are ready to be used, we need to set the configuration for the Locator. An example configuration file can be found in appendix A.1 and in file:

```
<ILS_HOME>\Locator\ils_locator_config.xml
```

Place a copy of this file in the WEB-INF directory of your Axis under Tomcat, for example in:

```
<TOMCAT_HOME>\webapps\axis\WEB-INF\
```

Note it should keep the same filename, i.e. `ils_locator_config.xml`, otherwise the software will not be able to access it. Now it is in place, edit it to adjust the settings to those appropriate to your requirements. The following sections describe the configuration settings in more detail.

4.2.2.1 Database Configuration

There are four settings for the configuration of the database, under the tag `Database` in the file `ils_locator_config.xml`:

- Driver – this should be the Java class of your JDBC driver, for example `"com.mysql.jdbc.Driver"`
- URL – the driver type, database type, URL and port on which the database accepts connections, for example `"jdbc:mysql://localhost:3306/"`
- Username – if access to the database server has been restricted you need to supply the username here
- Password - if access to the database server has been restricted you need to supply the password here

Note that this file is not encrypted so be aware that anyone who has access to it can see the username and password.

4.2.2.2 Policy Configuration

When an Initiator submits a request to the Locator, the Locator matches the job criteria in the request against its registered resources, and returns a list of resources available and capable of running the jobs. However, the exact number of resources returned depends not only on the availability but also on the resource count policy, which is set in the Locator configuration file under the `Policy` tag. The associated `brokerRatio` number for this policy is a multiplier used for determining how many resources to return in relation to the number requested. So, assuming no other restrictions such as resources being busy or unable to satisfy the job requirements, for n resources requested, $n * \text{brokerRatio}$ resources will be returned.

4.2.3 Testing the web services

We have provided a command line program for demonstrating that the Locator web services are up and running. Ensure that both Tomcat and your database server are running, and then type the following command:

```
java uk.ac.ed.epcc.pgpggrid.ILS.Locator.utils.ILSLocatorDemo
-url <URL>
```

Where `<URL>` should be replaced with the location of both the services, e.g.:

```
http://<YOUR_HOSTNAME>:<YOUR_PORT>/axis/services/
```

The on-screen output will indicate the progress of this command and describe the operations being performed. If any problems occur, then please check you have followed the previous steps correctly and review the Tomcat and Axis documentation for assistance. Remember also to check the Tomcat console output as well as the output from the command line.

4.3 Servent

The Servent and Initiator installations are much more straightforward than the Locator as they are web service clients and, as such, just require that the Axis libraries and the JAR file `<ILS_HOME>\pgpggrid_ils.jar` be visible in the `CLASSPATH` as described in section 4.1. Once this has been done the installation can be tested (assuming access to a Locator elsewhere) using the Locator demonstration code as described in section 4.2.3.

4.3.1 Servent Configuration

The Servent utilises a configuration file, an example of which is in appendix A.2 and also in the release at:

```
<ILS_HOME>\Servent\ils_servent_config.xml
```

There are six resource parameters that must be set for the Servent:

- `Hostname` – the URL of the resource e.g. `"yourmachine.yourdomain.com"` or the I.P. address, e.g. `"123.456.23.123"`
- `CPUCount` – the number of CPUs at the resource
- `Memory` – the memory available to each CPU in Megabytes
- `CPUSpeed` – the processor speed of each CPU in Gigahertz

- Bandwidth– the typical bandwidth between the Servent and the Locator in 1/seconds
- Benchmark– a value determined using the execution time for the supplied benchmark code (dimensionless)

The bandwidth figure is determined using code based on the system `ping` command. Ensure that the local installation of `ping` is visible at the command line (i.e. the `PATH` environment variable is appropriately set etc). The code is executed via the command below, where `LocatorHostAddress` is the URL of the Locator machine, for example `yourmachine.yourorg.com`:

```
java uk.ac.ed.epcc.pgpgrid.ILS.utils.ILSPing
-address LocatorHostAddress
```

Note that this figure is only representative of the bandwidth between the Servent and the Locator. It would be more useful to have a figure for the Servent to Initiator connection but this is not yet feasible.

Running the command above will produce on screen output that will include the average round trip time (RTT), usually in milliseconds, for the message packet. The bandwidth figure is then calculated as $1/RTT$ with *RTT expressed in seconds*. Note that the ILS software expects this value to be an integer.

The benchmark value is calculated using code from the Java Grande Forum [8] benchmark suites: in this case a ray tracer rendering code. The benchmark should be run several (say five) times on an unloaded machine so that an average figure can be determined for the score. It takes <30 seconds on a 2.0GHz Pentium IV processor. Use the total run time from the code output. Run the benchmark in directory `<ILS_HOME>\benchmark` via the command:

```
java -jar jgf.jar
```

This will produce a time in seconds that it took the benchmark to complete. Repeat and determine the average, and then we divide this figure by the time taken for the benchmark on a standard machine, which we will fix at 1 second. This then renders the benchmark value as a dimensionless figure inversely related to performance, that is, a higher value for this number implies that the machine will execute the code more slowly than a machine with a lower value. However, please note that the ray-tracing benchmark used here is merely representative of a particular type of application.

Once you have these calculated values edit the supplied configuration file or create your own and insert the parameters for your resource. Note that the name of this file does not affect execution of the code as it must be explicitly named in the command-line clients.

4.3.2 Servent Command Line Clients

To help demonstrate the Servent functionality we have provided a set of command line clients that interact with the Locator Registry web service. They all rely on your Servent configuration file as described in section 4.3.1, and referred to in the examples below as “`ConfigFile.xml`”. Also in the examples `RegistryURL` refers to the location of the Locator Registry web service which the Servent will interact with, for example “`http://localhost:8080/axis/services/ILSRegistry`”.

4.3.2.1 AddServent

This client registers your resource at the Locator. On execution the client contacts the Locator with your resource description parameters, and the Locator adds them to its database. Your

```
java uk.ac.ed.epcc.pgpgrid.ILS.Servent.ILSAddServentClient
-config ConfigFile.xml
-address RegistryURL
```

resource is then available to be matched to job requests from an Initiator. Run the client via the following command:

4.3.2.2 UpdateStatus

This client sets the number of processors *currently* available for running jobs at your resource. NumProcs below represents that number and is less than or equal to the total number of CPUs. The Locator receives this number and updates its database accordingly. So, for example, if you set the NumProcs to zero then your resource would not be available to be matched to job requests. The command is:

```
java
uk.ac.ed.epcc.pgpgrid.ILS.Servent.ILSUpdateStatusClient
-config ConfigFile.xml
-address RegistryURL
-numproc NumProc
```

4.3.2.3 RemoveServent

This client removes, or deregisters, your resource from the Locator database. Once executed the resource must be reregistered using the AddServent client (section 4.3.2.1) to make it available again for Initiator job matching. To temporarily remove your resource in the sense of making it unavailable for job matching the UpdateStatus client (section 4.3.2.2) should be used instead. Remove your resource via the command:

```
java
uk.ac.ed.epcc.pgpgrid.ILS.Servent.ILSRemoveServentClient
-config ConfigFile.xml
-address RegistryURL
```

4.4 Initiator

As mentioned in section 4.3, the installation of the Initiator just requires that the Axis libraries and the JAR file `<ILS_HOME>\pgpgrid_ils.jar` be visible in the CLASSPATH (see section 4.1). Once this has been done the installation can be tested (assuming access to a Locator elsewhere) using the Locator demonstration code as described in section 4.2.3.

4.4.1 Initiator Resource Request

The Initiator makes a request for resources using on the four resource request parameters:

- CPUCount – the number of CPUs required (N.B.: these are not necessarily on the same machine, i.e. they represent the number of single-processor jobs rather than one multi-processor job)
- Memory – a minimum acceptable value for the memory available on a single processor at the resource
- Bandwidth – the minimum acceptable bandwidth (N.B.: as discussed in section 4.3.1, this number will be measured against the bandwidth between the Locator and Servent, as at this stage the Initiator-Servent bandwidth is unavailable).
- Benchmark – the maximum acceptable benchmark value of the resource (that is, only resources with a benchmark value lower than this will be considered)

For the command line client below these request parameters are provided via an XML file, an example of which is in appendix A.3 and also in the file:

```
<ILS_HOME>\Initiator\ils_resource_request.xml
```

4.4.2 Initiator Command Line Client

To help demonstrate the Initiator functionality we have provided a command line client that interact with the Locator Request web service. It relies on your Resource Request file as described in section 4.4.1, and referred to in the example below as “RequestFile.xml”. Also in the example RequestURL refers to the location of the Locator Request web service, for example “http://localhost:8080/axis/services/ILSRequest”. Run the client via the following command:

```
java
uk.ac.ed.epcc.pgpgrid.ILS.Initiator.ILSRequestResourceClient
-request RequestFile.xml
-address RequestURL
```

The client contacts the Locator with your request parameters. The Locator then compares the available resources in its database to the parameters, and returns a list of resources that satisfy the criteria. The Locator policy (see section 4.2.2.2) as well as resource availability determines the number of resource descriptions returned. The client will list the returned resources to screen, or in the case if no resources will give the reason why.

5 Conclusions

We have described how to set up and install the ILS software. In addition we have provided within the release some command line clients to demonstrate the functionality of the ILS system. For further details on the software please see the associated Java documentation and the ILS User Guide [7].

6 References

1. PGPGrid project, <http://www.epcc.ed.ac.uk/~pgpgrid/>
2. ILS Use Cases, EPCC-IS-PGPGRID-UC-ILS
3. ILS Requirements and Survey, EPCC-IS-PGPGRID-REQ-ILS
4. Web Services W3C Note, <http://www.w3.org/TR/ws-arch/>
5. Axis, <http://ws.apache.org/axis/>
6. ILS High Level Design, EPCC-IS-PGPGRID-DES-ILS
7. ILS User Guide, EPCC-IS-PGPGRID-UG-ILS
8. Java Grande Forum, <http://www.javagrande.org/>

Appendix A

We append some example XML files for the Locator and Servent configuration, and for Initiator resource requests.

A.1 Example Locator Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) The University of Edinburgh, All Rights Reserved -->
<LocatorConfig>
  <!-- Database settings -->
  <!-- The driver class for the database, e.g.
       "com.mysql.jdbc.Driver" (String) -->
  <Database type="driver" value="com.mysql.jdbc.Driver"/>
  <!-- The driver type, database type and database port,
       e.g. "jdbc:mysql://localhost:3306/" (String) -->
  <Database type="url"
       value="jdbc:mysql://localhost:3306/" />
  <!-- The username for accessing the database (String) -->
  <Database type="username" value="" />
  <!-- The password for accessing the database (String) -->
  <Database type="password" value="" />

  <!-- Policy settings -->
  <!-- The ratio of resources returned to resources
       requested (double, min 1.0, max 10.0) -->
  <Policy type="brokerRatio" value="1.0" />
</LocatorConfig>
```

A.2 Example Servent Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) The University of Edinburgh, All Rights Reserved -->
<resources>
  <!-- Host IP address e.g. "mymachine.myorg.com" or
       "123.123.12.123" (String)-->
  <resource type="hostname" value="129.215.62.133"/>
  <!-- Total number of processors at the resource
       (Integer > 0) -->
  <resource type="cpucount" value="1"/>
  <!-- Memory (in Megabytes) for each processor
       (Integer > 0) -->
  <resource type="memory" value="640"/>
  <!-- CPU clock speed (in GigaHertz) of each
       processor (Double > 0) -->
  <resource type="cpuspeed" value="1.7"/>
  <!-- Bandwidth value (in 1/seconds) between the resource
       and the Locator (Integer > 0) -->
  <resource type="bandwidth" value="100"/>
  <!-- Benchmark value for the processor (Double > 0) -->
  <resource type="benchmark" value="30.0"/>
</resources>
```

A.3 Example Resource Request File

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) The University of Edinburgh, All Rights Reserved -->
<resources>
  <!-- Number of processors required (not necessarily on
        the same machine) for the job (Integer > 0) -->
  <resource type="cpucount" value="1"/>
  <!-- Minimum acceptable memory (in Megabytes) required at
        each processor for the job (Integer > 0) -->
  <resource type="memory" value="230"/>
  <!-- Minimum acceptable bandwidth (in 1/seconds) between
        resource and Locator (Integer > 0) -->
  <resource type="bandwidth" value="50"/>
  <!-- Maximum acceptable benchmark value - lower is better
        as it implies the processor is
        faster (Double > 0) -->
  <resource type="benchmark" value="50"/>
</resources>
```