



Mobile Command Center Admin Guide v. 1.0.1

Piotr Grabowski <piotrg@man.poznan.pl>
Bartek Lewandowski <bartekel@man.poznan.pl>

Table of Contents

Introduction	1
Mobile Command Center functionality	1
Job submission and control	2
Visualizations viewing	3
Message Box browsing	3
Auxiliary functionality	3
Mobile Command Center installation	4
Requirements	4
Installation steps	4
Message Box Webservice installation	6
Requirements	6
Installation steps	6
Message Box API installation	9
Requirements	9
Installation steps	10
Notification Webservice installation	11
Requirements	11
Installation steps	11
Summary	15
Glossary	15

Introduction

The main aim of this guide is to introduce functionalities provided by the GridLab Mobile Command Center (MCC) v. 1.0.1. In a nutshell, Mobile Command Center is a portlet that serves as a gateway for mobile devices, developed under the GridLab IST-2001-32133 project (www.gridlab.org [<http://gridlab.org>]), via which users have access to their grid resources from mobile devices like phones or PDAs.

Mobile Command Center functionality

MCC capabilities provided for end users can be divided into several groups according to functionalities provided by underlying services. The following groups can be distinguished:

- GridLab Resource Management System (GRMS) Job submission and control,
- Visualizations viewing,
- Message Box browsing,
- Auxiliary functionality.



The next sections will describe all these functionality groups and involved services in more details.

Job submission and control

The GridLab Resource Management System (GRMS) is an open source meta-scheduling system, based on dynamic resource selection, mapping and advanced scheduling methodology, combined with a feedback control architecture. It deals with dynamic Grid environment and resource management challenges, e.g. load-balancing among clusters, remote job control or file staging support. Therefore, the main goal of GRMS is to directly manage the whole process of remote job submissions to various batch queuing systems, clusters or resources. It has been designed as an independent set of core components for resource management processes, which can take advantage of various low-level Core Services and existing technologies. Finally, GRMS can be considered as a robust system which provides an abstraction of the complex grid infrastructure, as well as a toolbox which helps to form and adapt to distributing computing environments. GRMS is based on Globus Toolkit 2.X and uses its services deployed on resources. It means that GRMS provides job and resource management mechanisms on the top of Core Services. Moreover, GRMS supports the Grid Security Infrastructure by providing GSI-enabled Web Service interfaces for all clients, e.g. portals, mobile or command line clients or applications. Therefore, it can be integrated with other service-oriented grid middleware infrastructures. GRMS has been developed entirely in Java, and thus could be installed on many different kinds of operating systems and resources. One of the main assumptions for GRMS is to perform remote jobs control and management in the way that satisfies Users (Job Owners) and their applications requirements. All users requirements are expressed through XML-based resource specification documents, called GRMS Job Description, and sent to GRMS as SOAP requests over GSI transport layer connections. Simultaneously, Resource Administrators (Resource Owners) have full control over resources on which all jobs and operations will be performed by appropriate GRMS setup and installation. GRMS capabilities provided for end users can be divided into several groups according to GRMS functionality. The following groups can be distinguished: Job submission and control, listing jobs according to some criteria, managing jobs, getting information about jobs, getting a list of resources which meet user's requirements and criteria, managing notifications, auxiliary functionality. For further details please look at <http://www.gridlab.org/WorkPackages/wp-9>.

The most important part of the GRMS's functionality is the ability to submit a job to "the best" resource according to job resource requirements. In general, there are two possibilities: a user can specify "the best" resource in advance or "the best" resource will be chosen by GRMS. If the user specifies the resource in advance no scheduling and matching techniques will be used. Otherwise, GRMS will use a multi-criteria matchmaking algorithm to choose "the best" resource for the job. For job submission and control the XML-based document called GRMS Job Description has to be used by all clients. Using this document the user passes to GRMS all information needed by the system to execute the job (for example, the location of the executable, files that have to be staged in, arguments, environment variables, checkpoint files, etc.). The GRMS Job Description schema and examples are described and explained in the GRMS User Guide (see www.gridlab.org/workpackages/grms for details).

The following functionality of GRMS can be used for job submission and control from mobile device:

- `submitJob` – it is the main functionality of GRMS. Using it the user can submit the job described in a GRMS Job Description to be executed by GRMS. If the description is valid, GRMS puts the job into the queue and returns a globally unique job identifier (`GRMS_JOB_ID`) to the user, which unambiguously identifies the job in the system.
- `migrateJob` – this functionality allows the user to migrate the running job to a "better" resource (if such resource exists). The job is identified by the job identifier returned by the `submitJob` method. The whole process of job migration is relatively simple. First, the job is checkpointed on the resource, which is currently running on and then restarted on a new resource pointed by the user or chosen by GRMS. Note, that the migration process is performed by GRMS according to a new job description passed as the parameter to this method. If the new job description was not defined GRMS tries to perform the request basing on the job description passed during submission or a previous migration request. The migration would be possible only when a better resource was found and, the job was checkpointable (to be checkpointable the job has to either implement "checkpoint" web service interface and be registered in grms or be able in proper way to serve GAT checkpoint command sent by Mercury Service).
- `suspendJob` – using this functionality the user is able to suspend the running job. It means that the job will be



checkpointed and all checkpoint files/directories will be staged out. If a new job description is not defined the previous one will be used. Only checkpointable jobs can be suspended (see migrateJob functionality). This functionality is not implemented yet.

- resumeJob – this functionality resumes the execution of the job which was previously suspended. It is possible to define a new job description or use the previous one. This functionality is not implemented yet.
- cancelJob – this functionality allows the user to stop the running job. The difference between the suspendJob and cancelJob functionality is that the job is stopped (killed) by the system and does not create any checkpoint files/directories.

Visualizations viewing

Working together with the WP 8 Data Management and Visualization, WP12 Access for Mobile Users work package delivered a specialized service accessible from mobile user devices. WP12 users expressed the need to display scaled-down visualizations (mainly pictures) on their mobile phones or PDAs many times. Although large resolution images in many different formats can be easily accessed and displayed on desktop computers with high bandwidth networks, they are completely useless working with a mobile device. Small screens with low resolutions and colors depth together with intermittent, small bandwidth network access in case of mobile devices implies the need to develop software infrastructure that could scale down 'heavy' visualizations into formats more acceptable for small devices. The main idea of a new service is getting new versions of visualization appropriate to the device constraints and features. It is essential that the Visualization Service could change the file format of visualization - in case of mobile devices we are not interested in large dimension, not compressed files. The most suitable format would be PNG, therefore the service has to have a possibility to recode images into PNG files using dimensions of mobile device screen and colors as a base for new image dimensions. Taking into consideration all the above, a new method for mobile clients was added to the existing service interface. The parameters of this method are: source and output visualization placements, desired color depth, output picture size and cropping parameters. This service method can be called by any web service client and the output image can be accessed via HTTP(S) from any place and device. The call parameters should be set by user/developer according to device capabilities. This can be done automatically by MCC if the request for visualization is sent indirectly (except the cropping properties which have to be set by the user - see Mobile Client Introduction). Two additional methods were added to the Visualization service interface after service deployment and first successful tests. One of them allows to send parameters for cropping as percentage values of the whole image size (this even allows the client not to know the source image dimensions). The second additional method is used to delete the previously obtained images from the server.

Message Box browsing

The usage scenario for mobile devices requires that the GridLab Access for Mobile Users workpackage developers should provide a new service that could send and store different kinds of messages. To fulfill the scenarios requirements we developed 3 applications: the notification server - Message Box, Mobile Client - Message Reader, and the Dedicated server for the Message Box - the Message Box Server (HTTP/HTTPS). The Message Box application was designed as Java API that can be used from an external package like portlets or a dedicated servlet server. The Web Service extension for Message Box API was added in the last months of the year 2003. The Message Box database was improved to contain also the user and user profile data. This allowed us to have unique identifiers of users regardless of the client that uses our service which is now a stand alone service and does not need to be deployed in other application/portal environment. The main purpose of the Message Box Service is to store users notification messages in the folder structured repositories. These messages can be also send to the user using different technologies. It is possible to send messages as Email, SMS and in the future as Fax. All messages can be retrieved from Message Box using the Message Box Service Interface. One of the client of the Message Box is the mobile Command Center which is used as a gateway for GridLab Mobile Client. Therefore, GMC can access user folders and messages stored in Message Box.

Auxiliary functionality

The functionality listed below is a part of the GridSphere framework for user login and profile.



- user login - the user can login into GridSphere with this functionality
- user logout - the user can logout from GridSphere with this functionality
- user credential usage - this is done automatically without user action - user credential stored in GridPortlets credential repository is used to connect to external grid-services (like GRMS or MessageBox)
- user data checking - the user can check his/her profile data using this functionality. It is also good way to check the gateway availability.

Mobile Command Center installation

Requirements

These are minimal software requirements your environment must meet to MCC work properly:

- Java 1.4.2 (from <http://java.sun.com>) (see the GridSphere requirements)
- Jakarta Ant >= 1.6+ (from <http://www.apache.org/dist/ant/binaries> [`<?xml version="1.0"?> <ns:clipboard xmlns:ns="http://www.xmlmind.com/xmleditor/namespace/clipboard" ><ulink url="???" >http://www.apache.org/dist/ant/binaries</ulink ></ns:clipboard >`]) (see the GridSphere requirements)
- Jakarta Tomcat 4 or 5 (from <http://www.apache.org/dist/jakarta/tomcat>) (see the GridSphere requirements)
- GridSphere installed and deployed (see <http://www.gridisphere.org> for details).
- GridPortlets installed and deployed in the same GridSphere container (see <http://www.gridisphere.org> for details).
- MCC portlet sources downloaded either from CVS or as a compressed archive

Installation steps

1. Place the MCC portlet application in the GridSphere projects directory of your GridSphere source directory (unzip if necessary)
2. Type "ant install" in project root directory

```
[yourlogin@yourhost mobilecommandcenter]$ ant install
```
3. You should see "BUILD SUCCESSFUL"
4. Start the GridSphere and go with your browser to your GridSphere page e.g.:

```
http://localhost:8080/gridisphere/gridisphere
```
5. Login as root, go to the GridSphere administration tab and deploy the "mobilecommandcenter" as portlet, you should see the "mobilecommandcenter" entry in "Portlet web applications" section
6. You should see the "mccp" servlet is already working in "Non-portlet web applications" section (the Mobile Command Center servlet context is automatically added to webapps dir during "ant install" procedure)
7. Now, you can configure your group membership settings ("Welcome" tab) - check the checkbox near the "mobilecommandcenter" entry and push "save" button - you should see the "Mobile Command Center" tab

appearing as the last tab in your collection.

8. Go there and you should see the Mobile Command Center like on the next figure

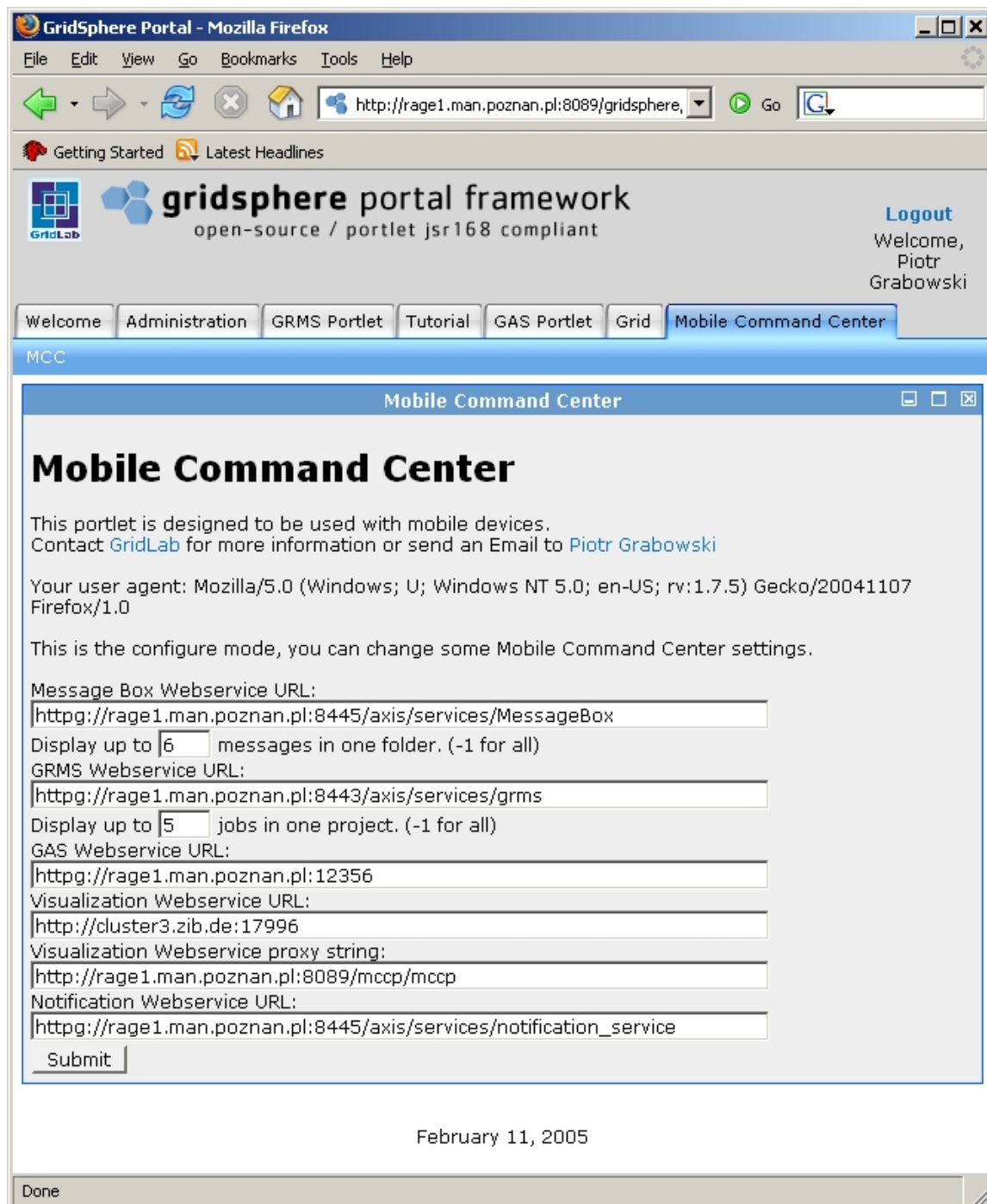


Figure 1.



9. Now, you can save your Mobile Command Center options to use (like Message Box gridservice placement, GRMS service placement, the amount of messages/job descriptions that is send to mobile device or Visualization or Notification Webservice URLs).

Important

The Mobile Command Center Portlet will show editable version of portlet presentation only if the logged in user has admin rights.

Make sure that your version of the GridPortlets did not copied into Tomcat's shared/lib directory any old jars containing MessageBox classes (org.gridlab.messaging.xxxxx), in this case delete it or replace with the newest version of MessageBoxWebservice.jar.

Message Box Webservice installation

Requirements

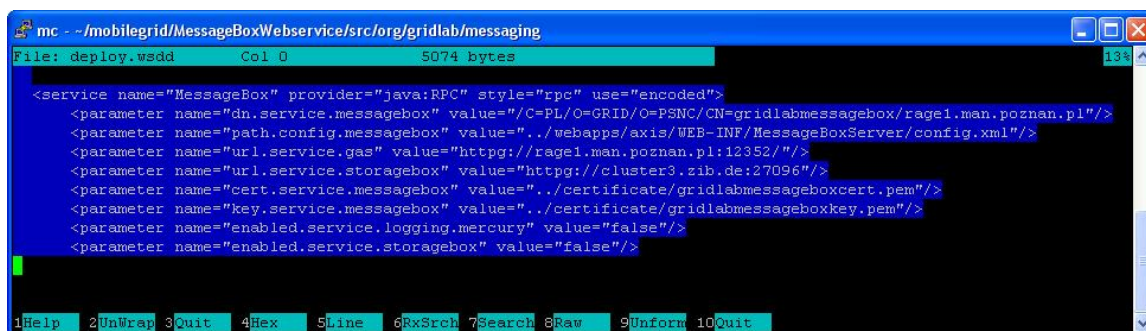
These are minimal software requirements your environment must meet to Message Box work properly:

- Java 1.4.2 (from <http://java.sun.com>)
- Jakarta Ant >= 1.6+ (from <http://www.apache.org/dist/ant/binaries>)
- Jakarta Tomcat 4 or 5 (from <http://www.apache.org/dist/jakarta/tomcat>)
- Apache Axis 1.1 (from <http://ws.apache.org/axis>)
- MySQL server >=3.5 (from <http://mysql.com>)
- Message Box Service sources downloaded either from CVS or as a compressed archive

Installation steps

1. Unzip or get from CVS the whole project tree
2. Go into root dir of the project ("MessageBoxWebservice")
3. Type "ant" in project root directory

```
[yourlogin@yourhost MessageBoxWebservice]$ ant
```
4. You should see "BUILD SUCCESSFUL"
5. Go into "src/org/gridlab/messaging" dir
6. Check the settings in "deploy.wsdd" file - change parameters to appropriate values (this values are stored in Axis server-config.wsdd files an can be used to change Message Box Service settings without service redeploying)



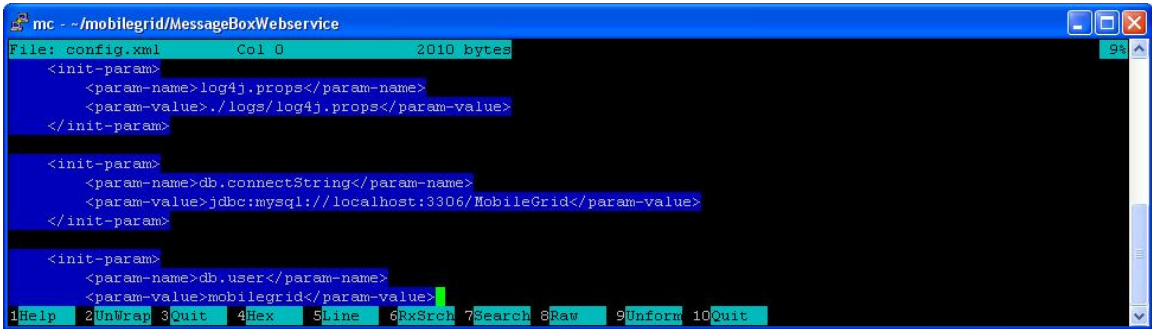
```

mc - - /mobilegrid/MessageBoxWebservice/src/org/gridlab/messaging
File: deploy.wsdd Col 0 5074 bytes 134
<service name="MessageBox" provider="java:RPC" style="rpc" use="encoded">
  <parameter name="dn.service.messagebox" value="/C=PL/O=GRID/O=PSNC/CN=gridlabmessagebox/rage1.man.poznan.pl"/>
  <parameter name="path.config.messagebox" value="../webapps/axis/WEB-INF/MessageBoxServer/config.xml"/>
  <parameter name="url.service.gas" value="http://rage1.man.poznan.pl:12352/">
  <parameter name="url.service.storagebox" value="http://cluster3.zib.de:27096"/>
  <parameter name="cert.service.messagebox" value="../certificate/gridlabmessageboxcert.pem"/>
  <parameter name="key.service.messagebox" value="../certificate/gridlabmessageboxkey.pem"/>
  <parameter name="enabled.service.logging.mercury" value="false"/>
  <parameter name="enabled.service.storagebox" value="false"/>

```

Figure 2.

- | | |
|---------------------------------|---|
| dn.service.messagebox | The distinguish name of your Message Box GridService. |
| path.config.messagebox | Path to MessageBox API config.xml file (can be relative from Tomcat's "bin" dir) |
| url.service.gas | The URL to GAS gridservice (see GridLab Security pages) |
| url.service.storagebox | The URL to Storage Box gridservice (see GridLab Data pages) |
| cert.service.messagebox | Path to the service certificate file |
| key.service.messagebox | Path to the service private key file |
| enabled.service.logging.mercury | Enabling/Disabling Mercury logging (see GridLab Monitoring pages). The value can be "true" or "false". |
| enabled.service.storagebox | Enabling/Disabling Storage Box forwarding (see GridLab Monitoring pages). The value can be "true" or "false". |
7. Check the settings in "deploy.sh" or "deploy.bat" files and launch it
 8. Now, the webservice should be deployed to Axis - look for "MessageBox" service section in Axis server-config.wsdd file. If deployment failed for some reason you may just copy the content of deploy.wsdd file into server-config.wsdd file.
 9. Go to the project root dir.
 10. Change the settings for Message Box API in "config.xml" file and copy it to dir pointed as service "path.config.messagebox" parameter. The complete list of settings can be found in Message Box API Admin Guide section



```
mc - -/mobilegrid/MessageBoxWebservice
File: config.xml Col 0 2010 bytes
<init-param>
  <param-name>log4j.props</param-name>
  <param-value>./logs/log4j.props</param-value>
</init-param>

<init-param>
  <param-name>db.connectString</param-name>
  <param-value>jdbc:mysql://localhost:3306/MobileGrid</param-value>
</init-param>

<init-param>
  <param-name>db.user</param-name>
  <param-value>mobilegrid</param-value>
</init-param>
1Help 2UnWrap 3Quit 4Hex 5Line 6ExSrch 7Search 8Raw 9Unform 10Quit
```

Figure 3.

11. If MessageBox API was not previously deployed you need to change the settings in "deployMessageBoxToMySQL.sh" file according to database settings from the "config.xml" file and launch it.
12. Restart tomcat
13. Check with browser the Axis servlet for MessageBox existence

<http://localhost:8080/axis/servlet/AxisServlet>

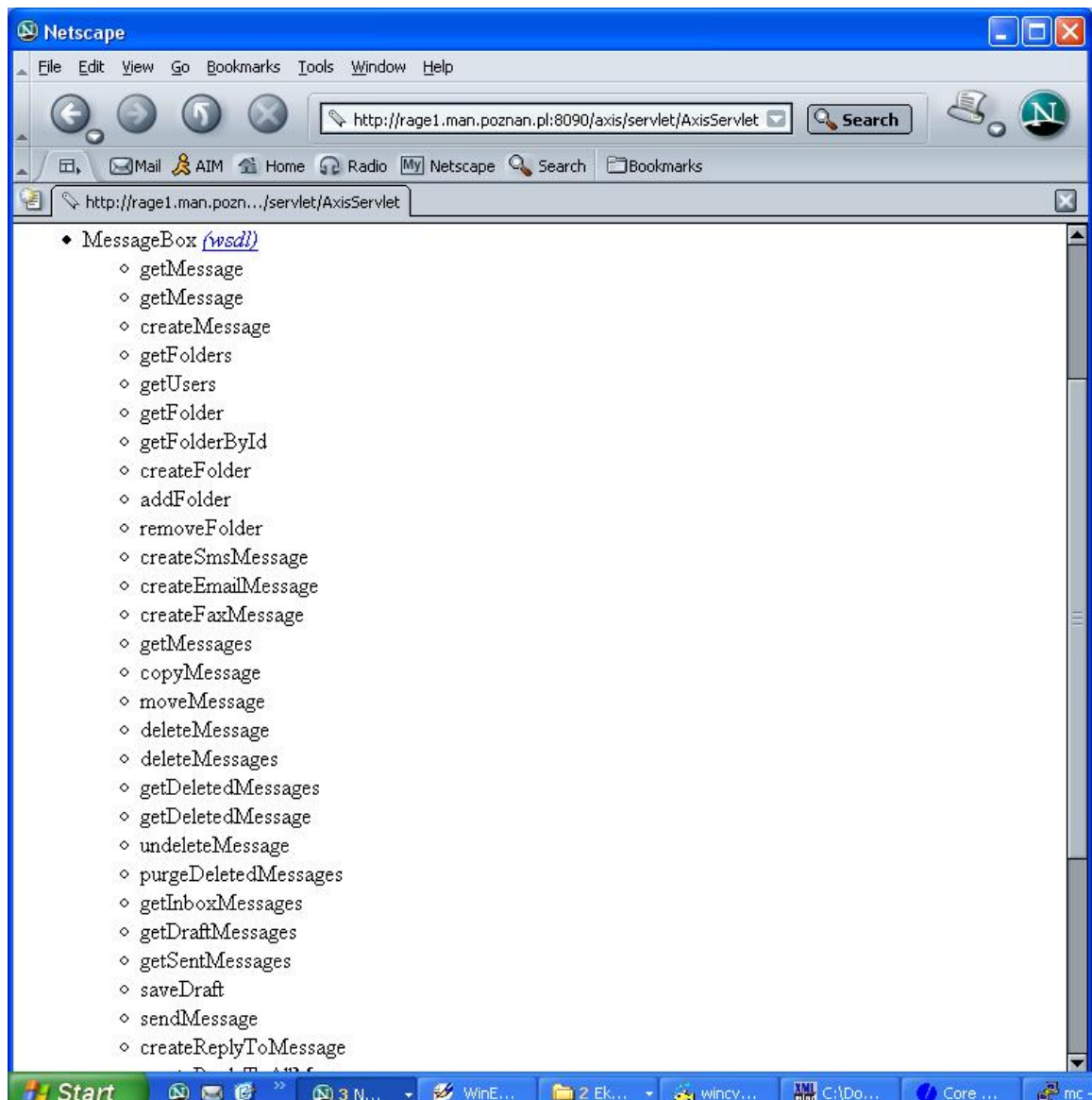


Figure 4.

14. Now, you can change your Mobile Command Center options to use this Message Box Service instance or build and start one of the Message Box clients to check if the service is properly installed.

Message Box API installation

Requirements

These are minimal software requirements your environment must meet to Message Box work properly:



- Java 1.4.2 (from <http://java.sun.com>)
- Jakarta Ant >= 1.6+ (from <http://www.apache.org/dist/ant/binaries> [`<?xml version="1.0"?> <ns:clipboard xmlns:ns="http://www.xmlmind.com/xmleditor/namespace/clipboard" ><ulink url="???" >http://www.apache.org/dist/ant/binaries</ulink ></ns:clipboard >`])
- MySQL server >=3.5 (from <http://mysql.com>)
- Message Box API sources downloaded either from CVS or as a compressed archive

Installation steps

1. Unzip or get from CVS the whole project tree
2. Go into root dir of the project ("MessageBox")
3. Type "ant" in project root directory
`[yourlogin@yourhost MessageBox]$ ant`
4. You should see "BUILD SUCCESSFUL"
5. Check the settings in "config.xml" file - change parameters to appropriate values

```
mc - ~/mobilegrid/MessageBoxWebservice
File: config.xml Col 0 2010 bytes
<init-param>
  <param-name>log4j.props</param-name>
  <param-value>./logs/log4j.props</param-value>
</init-param>
<init-param>
  <param-name>db.connectString</param-name>
  <param-value>jdbc:mysql://localhost:3306/MobileGrid</param-value>
</init-param>
<init-param>
  <param-name>db.user</param-name>
  <param-value>mobilegrid</param-value>
</init-param>
1Help 2UnWrap 3Quit 4Hex 5Line 6RxSrch 7Search 8Raw 9Uniform 10Quit
```

Figure 5.

log4j.props

The path to the log4j properties file to be used by Message Box API. Can be relative to the application launch (prompt) directory (if the Message Box API is used within Message Box Service the launch dir will be the Tomcat's "bin" dir)

db.connectString

The full database connection string (with database name obligatory)

db.user

The database user to work with

db.password

The database user password to work with

db.driverClassName



- | | |
|--------------------------|---|
| | The database driver class name to be loaded by the API |
| mail.host.address | The address of SMTP server to be used by Email sending module |
| mail.host.user | The user name for authenticated connection with SMTP server. |
| mail.host.pass | The user password for authenticated connection with SMTP server. |
| mail.reply.title.begin | The default beginning of Email title if the Email is created in reply to other one. |
| mail.reply.wrote | The default beginning of Email body if the Email is created in reply to other one. |
| mail.forward.title.begin | The default beginning of Email title if the Email is created as forward to other one. |
| mail.forward.wrote | The default beginning of Email body if the Email is created as forward to other one. |
6. If MessageBox API was not previously deployed you need to change the settings in "deployMessageBoxToMySQL.sh" file according to database settings from the "config.xml" file and launch it. It will add the necessary database and database tables
 7. Now, you can change your Message Box Service (if any) options to use this Message Box API instance or start one of the Message Box clients to check if the API is properly installed.

Notification Webservice installation

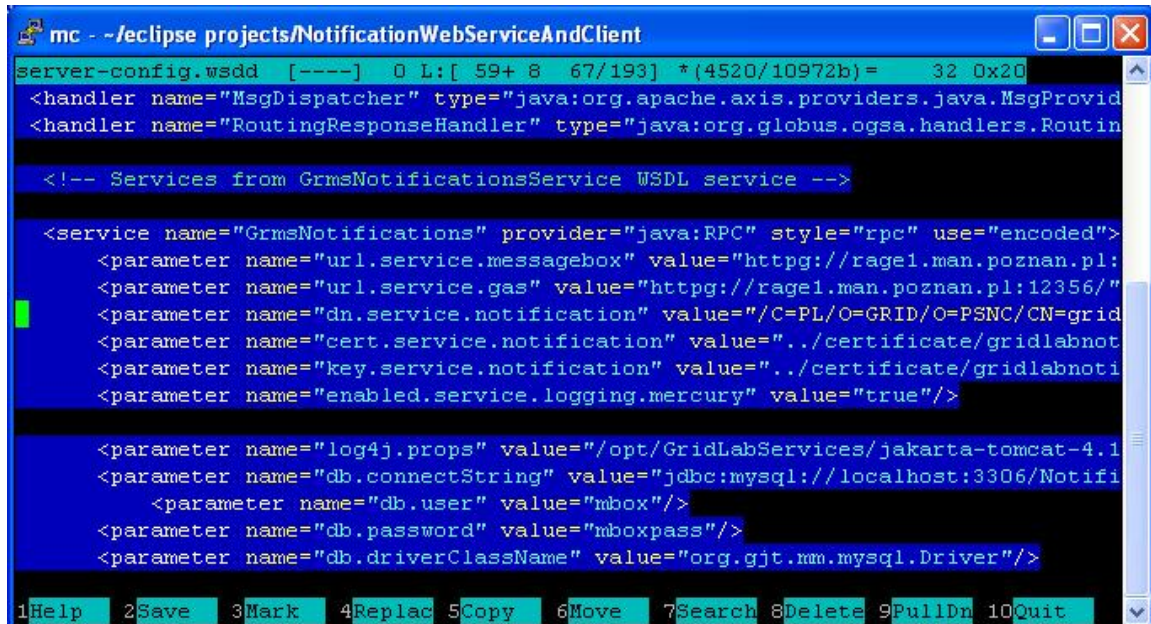
Requirements

These are minimal software requirements your environment must meet to Notifications work properly:

- Java 1.4.2 (from <http://java.sun.com>)
- Jakarta Ant >= 1.6+ (from <http://www.apache.org/dist/ant/binaries>)
- Jakarta Tomcat 4 or 5 (from <http://www.apache.org/dist/jakarta/tomcat>)
- Apache Axis 1.1 (from <http://ws.apache.org/axis>)
- MySQL server >=3.5 (from <http://mysql.com>)
- Notification Service sources downloaded either from CVS or as a compressed archive

Installation steps

1. Unzip or get from CVS the whole project tree
2. Go into root dir of the project ("NotificationWebServiceAndClient")
3. Type "ant" in project root directory
`[yourlogin@yourhost NotificationWebServiceAndClient]$ ant`
4. You should see "BUILD SUCCESSFUL"
5. Go into "src\grms\gl_services\grms_notification\stub dir"
6. Check the settings in "deploy.wsdd" file - change parameters to appropriate values (this values are stored in Axis server-config.wsdd file and can be used to change GRMS Notification Service settings without service redeploying)



```
server-config.wsdd [-----] 0 L:[ 59+ 8 67/193] *(4520/10972b)= 32 0x20
<handler name="MsgDispatcher" type="java:org.apache.axis.providers.java.MsgProvid
<handler name="RoutingResponseHandler" type="java:org.globus.ogsa.handlers.Routin

<!-- Services from GrmsNotificationsService WSDL service -->

<service name="GrmsNotifications" provider="java:RPC" style="rpc" use="encoded">
  <parameter name="url.service.messagebox" value="http://rage1.man.poznan.pl:
  <parameter name="url.service.gas" value="http://rage1.man.poznan.pl:12356/"
  <parameter name="dn.service.notification" value="/C=PL/O=GRID/O=PSMC/CN=grid
  <parameter name="cert.service.notification" value="./certificate/gridlabnot
  <parameter name="key.service.notification" value="./certificate/gridlabnoti
  <parameter name="enabled.service.logging.mercury" value="true"/>

  <parameter name="log4j.props" value="/opt/GridLabServices/jakarta-tomcat-4.1
  <parameter name="db.connectString" value="jdbc:mysql://localhost:3306/Notifi
    <parameter name="db.user" value="mbox"/>
  <parameter name="db.password" value="mboxpass"/>
  <parameter name="db.driverClassName" value="org.gjt.mm.mysql.Driver"/>

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Figure 6.

<code>dn.service.notifications</code>	The distinguish name of your Notification GridService.
<code>url.service.gas</code>	The URL to GAS gridservice (see GridLab Security pages)
<code>url.service.messagebox</code>	The URL to Message Box gridservice where the notifications are stored as messages and actually send to the users
<code>cert.service.notification</code>	Path to the service certificate file
<code>key.service.notification</code>	Path to the service private key file



- | | |
|--|--|
| <code>enabled.service.logging.mercury</code> | Enabling/Disabling Mercury logging (see GridLab Monitoring pages). The value can be "true" or "false". |
| <code>log4j.props</code> | The path to the log4j properties file to be used. |
| <code>db.connectString</code> | The full database connection string (with database name obligatory) |
| <code>db.user</code> | The database user to work with |
| <code>db.password</code> | The database user password to work with |
| <code>db.driverClassName</code> | The database driver class name to be loaded by the service |
7. Follow the instruction from `.wsdd` file to deploy the service to `Axis server-config.wsdd` file or just copy the service section content into `server-config.wsdd` file.
 8. Repeat steps 5-7 in `src\grms\gl_services\notifications\stub` directory for non-GRMS part of service.
 9. If you want to deploy also old version of notifications (compliant with GRMS 1.9.3 and older): repeat steps 5-7 in `src\gridlab\notifications\stub` directory
 10. Now, the webservice should be deployed to Axis - look for "Notifications" and "GrmsNotifications" service sections in `Axis server-config.wsdd` file. If deployment failed for some reason you may just copy the content of `deploy.wsdd` files into `server-config.wsdd` file.
 11. Go to the project root dir.
 12. If Notifications was not previously deployed you need to change the settings in "deployNotificationsToMySQL.sh" file according to database settings from the "server-config.wsdd" file and launch it.
 13. Restart tomcat

14. Check with browser the Axis servlet for Notifications existence

`http://localhost:8080/axis/servlet/AxisServlet`

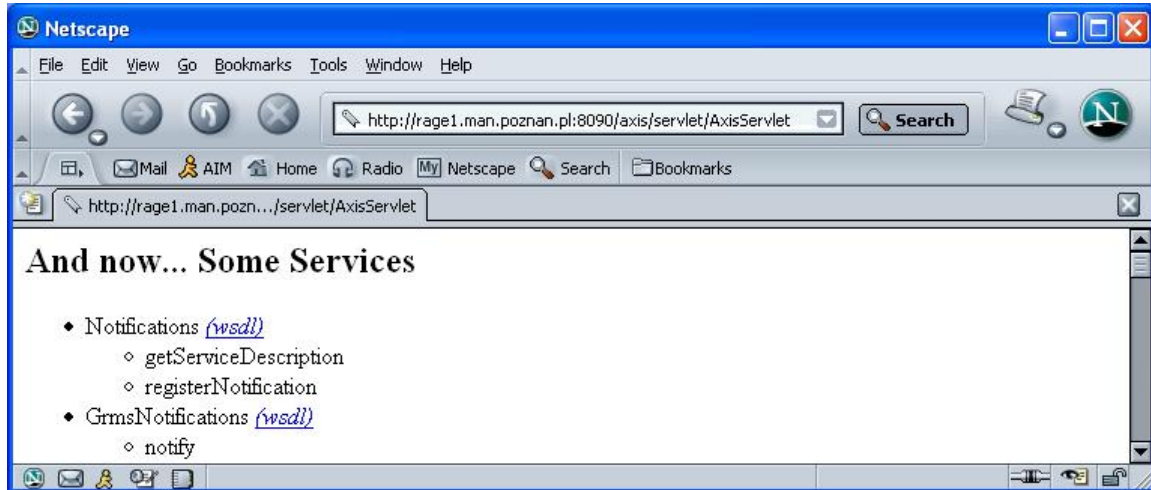


Figure 7.

15. Now, you can change your Mobile Command Center options to use this Notifications Service instance or start one of the Notification clients to check if the service is properly installed. The view of GrmsAWTNotificationClient is on the next figure.

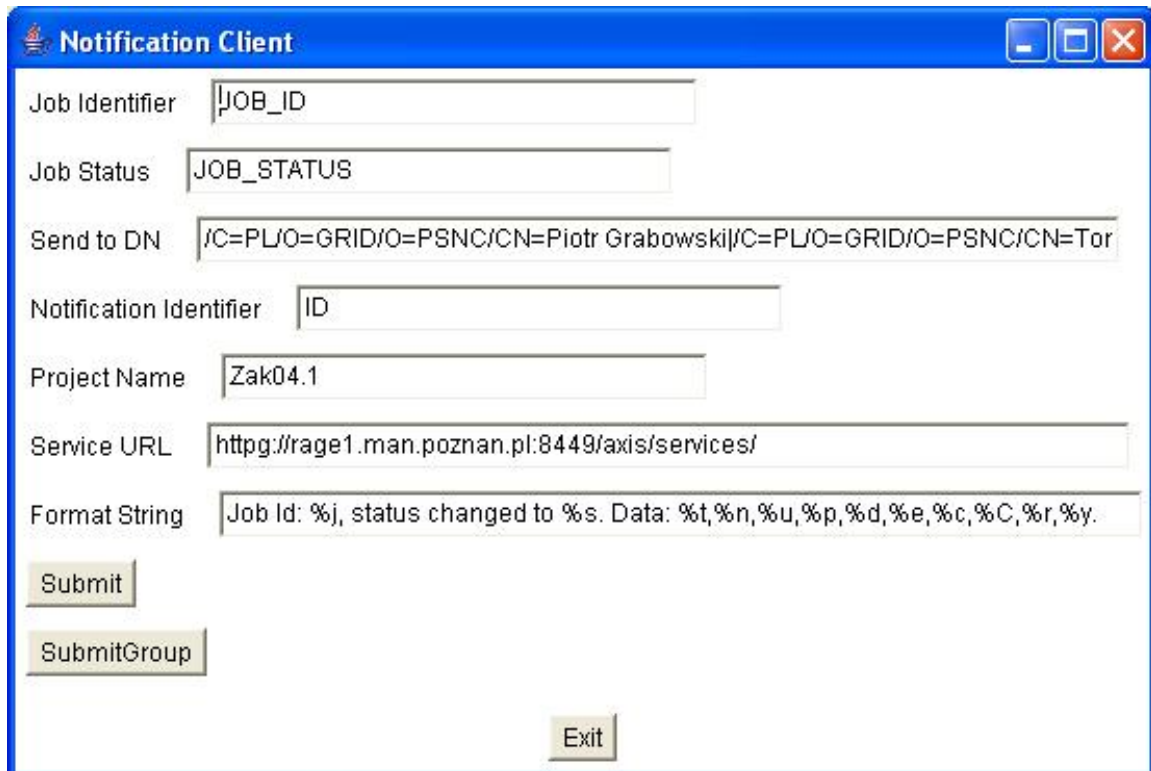


Figure 8.



Summary

This document presents The GridLab Mobile Command Center functionalities, brief introduction to underlying services like Message Box, GridLab Resource Management System or Visualization Service for Mobiles, and detailed, step by step instructions on how to install and use the MCC. In the last sections this document also explains how to install and administer GridLab Mobile Services. If you find any difficulties or problems, please contact <piotrg@man.poznan.pl> or <mobile@gridlab.org>.

Glossary

Basic definitions

Midlet	Java 2 Micro Edition application.
MCC	Mobile Command Center - portlet acting as a gateway to the grid for mobile devices.
GMC	GridLab Mobile Client - J2ME MIDP 1.0 midlet which is a client for MCC. It allows the user of mobile device to access his/her grid resources via the MCC gateway.
MIDP	SUN: The Mobile Information Device Profile (MIDP), when combined with the Connected Limited Device Configuration (CLDC), is the Java runtime environment for today's most popular compact mobile information devices, such as cell phones and mainstream PDAs.
J2ME	SUN: The Micro Edition of the Java 2 Platform provides an application environment that specifically addresses the needs of commodities in the vast and rapidly growing consumer and embedded space, including mobile phones, pagers, personal digital assistants, set-top boxes, and vehicle telematics systems.
CLDC	SUN: The Connected Limited Device Configuration (CLDC) defines the base set of application programming interfaces and a virtual machine for resource-constrained devices like mobile phones, pagers, and mainstream personal digital assistants.
GRMS	GridLab Resource Management System (GRMS) is an open source meta-scheduling system, developed under the GridLab Project, which allows developers to build and deploy resource management systems for large scale distributed computing infrastructures.
GridSphere	Grid portal development framework developed by GridLab portal team which provides an open-source portlet based Web portal that offers support for developing portlets and customizing websites online.
PDA	Personal Digital Assistant - a handheld device that combines computing, telephone/fax, Internet and networking features.