# GNDMS Generation N Data Management System Documentation Bundle

Stefan Plantikow and Maik Jorra
Zuse Institute Berlin (ZIB)

4.8.2010

## Contents

# Generation N Data Management System

The Generation N Data Management System (**GNDMS**) is a set of Globus Toolkit 4 WSRF services and associated tools for distributed grid data management based on staging and co-scheduling. It abstracts from data sources via a data integration layer and provides logical names, data transfers via GridFTP, proper handling of GSI certificate delegation and workspace management.

Besides data management functionality the implementation provides components for remote logging, run-time reconfiguration, persistence, and failover beyond what is available from Globus Toolkit 4.

Originally, GNDMS was written and deployed for the data management needs of the Collaborative Climate Community Data and Processing Grid (C3-Grid) and is now being used in the Plasma-Technologie-Grid (PT-Grid) as part of the German D-Grid grid computing initiative. Nevertheless, the implementation is flexible and has been written for reuse by other grid projects with similiar data management requirements. Core components may be of use to developers of non-data management GT4 services as well.

## Impressum (DE)


```
Konrad-Zuse-Zentrum fuer Informationstechnik Berlin
Takustrasse 7
D-14195 Berlin-Dahlem

GERMANY

Tel  +49 30 84185-0
Fax  +49 30 84185-125
Web  http://www.zib.de
```


Das Konrad-Zuse-Zentrum fuer Informationstechnik Berlin ist eine Anstalt des oeffentlichen Rechts.

### Rechtliche Grundlagen

Gesetz ueber das Zentrum fuer Informationstechnik (ZInfG)

Trotz sorgfaeltiger inhaltlicher Kontrolle uebernehmen wir keine Haftung fuer die Inhalte.

Sofern innerhalb unseres Internetangebotes die Moeglichkeit zur Eingabe persoenlicher oder geschaeftlicher Daten (Emailadressen, Namen, Anschriften usw.) besteht, so erfolgt die Preisgabe dieser Daten seitens des Nutzers auf ausdruecklich freiwilliger Basis.

## Impressum (EN)

Konrad-Zuse-Zentrum fuer Informationstechnik Berlin
Takustrasse 7
D-14195 Berlin-Dahlem

GERMANY

Tel  +49 30 84185-0
Fax  +49 30 84185-125
Web  http://www.zib.de

ZIB was founded as an institution under public law.

### Legal Grounds

Gesetz ueber das Zentrum fuer Informationstechnik (ZInfG)

Although we carefully revise all texts we assume no liability for the contents of external links for which solely the external provider is responsible.

In case our Internet offer provides the option to enter personal or business data (e.g. e-mail addresses, names, addresses), it is deemed to be agreed that the user expressly voluntarily reveals such data.

# GNDMS Installation Guide

This is the Installation Guide for the Generation N Data Management System.

## Prerequisites

In order to build or install GNDMS, the following prerequisites need to be fulfilled

### Prepare your local software installation

Install the Java 2 SE Development Toolkit Version 1.6

Please Install the Java 2 SE Development Toolkit Version 1.6.

For compiling the services, please make sure that `$JAVA_HOME` points to this version and that this is also the version that is in your `$PATH`. Naturally, this should be the same version than the one you use(d) for building and running globus and ant.

Install Apache Ant 1.7

Please install Apache Ant 1.7 and set `$ANT_HOME`, add it to your environment, and add `$ANT_HOME/bin` to your `$PATH`

**NOTE** *Using 1.8 might cause trouble on Linux, YMMV*

Install local UNIX software

In order to install GNDMS, please make sure you have installed the following software

- openssl
- curl
- rsync

Additionally, it is expected that your UNIX provides the following shell tools: hostname, which, bash

Install Globus Toolit 4.0.8

Please download and make a full installation of Globus Toolkit 4.0.8

**NOTE** *To be precise, GNDMS doesn't use CAS and RLS, everything else needs to be there. However, due to the way the GT4 build system works, we suggest you just install everything.*

**NOTE** *If you want to cut down the build, try* `./configure --prefix=/opt/gt-4.0.8 --with-flavor=gcc32dbg --disable-rls --disable-tests --disable-wstests --disable-drs` *(or* `--flavor=gcc64dbg` *on 64-Bit Linux or Mac OS X)*

- Setup working host and user certificates (You can build without)
- Set `$GLOBUS_LOCATION` and add it to your environment
- Life gets easier by putting `source $GLOBUS_LOCATION/etc/globus-user-env.sh` and `source $GLOBUS_LOCATION/etc/globus-devel-env.sh` in `$HOME/.profile` when working with Globus Toolkit
- We strongly suggest that you create a tarball of your fresh installation of globus toolkit for backup purposes. This allows you to rollback later and try again in case something goes wrong.

In the following it will be assumed that globus is run by the user `globus` which you will have created during the installation of Globus Toolkit.

Optionally Configure Globus Toolkit Logging

This step is optional but highly recommended.

To configure the Globus Container to generate substantially more log messages for easier debugging, please add

```
log4j.category.de.zib=DEBUG
```

to `$GLOBUS_LOCATION/container-log4j.properties`

For even more log information, please change the line that starts with

`log4j.appender.A1.layout.ConversionPattern=` to

```
log4j.appender.A1.layout.ConversionPattern=%d{ISO8601} %-5p %c{2} [%t,%M:%L] <%x> %m%n
```

in the same file.

**ATTENTION** *The default globus log file in `$GLOBUS_LOCATION/var/container.log` gets installed with very liberal file permisssions. You might want to `chmod 0640 $GLOBUS_LOCATION/var/container.log` for security reasons.*

Now it's time to start installing GNDMS.

### Preparation of GNDMS Software

**ATTENTION** *The following steps need to be executed as the `globus` user that runs the servlet container of your installation of Globus Toolkit.*

Download and Unpack GNDMS

Download either an official GNDMS distribution package and unpack it or get the current development version from github at:

`http://github.com/zibhub/GNDMS/downloads`

Please set `$GNDMS_SOURCE` to the root directory of your GNDMS distribution (The directory that contains `Buildfile`) and add `$GNDMS_SOURCE/bin` to your `$PATH`

Additionally, please set the following environment variables as specified below

- `$GNDMS_SHARED` to `$GLOBUS_LOCATION/etc/gndms_shared`
- `GNDMS_MONI_CONFIG` to `$GNDMS_SHARED/monitor.properties`

After this step, there should be no further need to adjust your environment. Please consult `$GNDMS_SOURCE/example.profile` for an example of a properly configured environment.

Optionally Install Apache buildr 1.4.1 locally

This step is optional.

GNDMS is built and installed using Apache buildr. A pre-packaged version of buildr is included with GNDMS and can be executed by running `$GNDMS_SOURCE/bin/gndms-buildr`. However, if you prever to install buildr locally, please

- Install Ruby 1.8
- Install buildr by executing `gem install buildr`

This guide assumes the usage of the pre-packaged version of buildr.

## Installation and Deployment from Distribution Package

This section describes the actual installation of the GNDMS software into the Globus Container. It requires that your system has been prepared as described in the previous section. Again the following steps should be executed by the `globus` user.

- Please enter `$GNDMS_SOURCE` and exeucte `gndms-buildr install-distribution`

This will

- Download and install required software dependencies into `$GLOBUS_LOCATION/lib`.

  **Please consult `$GNDMS_SOURCE/doc/licensing` for details on licensing conditions of 3rd party software components used by the GNDMS package.**
- Build API Documentation (Javadocs) in `$GNDMS_SOURCE/doc/api`
- and finally install the globus packages (gar-files)
- (Re)start the globus container with `globus-start-container-detached` and check `$GLOBUS_LOCATION/var/container.log` If everything goes right and you enabled additional logging as described in the previous section, the output should contain output like

  ```
  ============================================
  GNDMS RELEASE: Generation N Data Management System VERSION: 0.3
  "Rob" GNDMS BUILD: built-at: Wed Jul 21 11:14:01 +0200 2010
  built-by: mjorra@csr-pc35
  ============================================
  Container home directory is '/opt/gt-current'
  ```

(In the case of an error, you may want to compare with a full startup log).

- *After having checked succesful startup by looking at the logs*, fix the permissions of database files by executing

  gndms-buildr fix-permissions

**ATTENTION** *Skipping this step may cause leaking of sensitive information to local UNIX users*

At this point the GNDMS software has been succesfully installed. Next, we'll describe how it may be configured for actual use.

## Gridconfiguration of GNDMS Software

GNDMS is configured via a builtin monitoring shell that accesses and modifies the configuration in the database.

If you did a fresh installation, the monitoring shell will have been enabled temporarily at this point and you may just proceed. Otherwise you need to enable in manually as described in the following section.

### Enabling the Monitoring Shell Manually

To enable the monitor shell manually, after having startet the globus container with deployed GNDMS at least once (as described in the previous section), please edit `$GNDMS_MONI_CONFIG` such that `monitor.enabled` is set to `true` and either wait until GNDMS picks up the new configuration or restart the globus container.

The monitoring shell will be running now. You have nearly finished the installation at this point. All that is left to do, is to actually configure GNDMS for the chosen community grid platform.

**NOTE** *The shell is accessed via localhosts network interface and protected with a clear-text password only. Do not make the monitoring shell accessible via unsecure networks.*

### Disabling the Monitoring Shell

To disable the monitor shell, please edit `$GNDMS_MONI_CONFIG` such that *both* `monitor.enabled` and `monitor.noShutdownIfRunning` are set to `false`. Now, either wait until the new configuration gets activated or just restart the globus container manually.

### Configuring your Grid

Currently, there are specialized build targets for the setup of some D-Grid projects directly in the `Buildfile`.

**PT-Grid Setup & Configuration** : Edit `$GNDMS_SOURCE/scripts/ptgrid/setup-resource.sh` and execute `gndms-buildr ptgrid-setubdb`

**PT-Grid Quick Test** : Follow the setup instructions in the testing section below and execute `gndms-buildr ptgrid-test`

Additionally, please consult the documentation for the respective community grid platform.

**NOTE** *In case of failure during setup, please execute* `gndms-buildr kill-db` *and try again.*

### Finalize installation

Please edit `$GLOBUS_LOCATION/etc/gndms_shared/monitor.properties` and set `monitor.enabled=false` and `monitor.noShutdownIfRunning=false`. This will disable the monitor shell after `monitor.configRefreshCycle` ms (defaults to 17 seconds). Alternatively, just restart the globus container.

**Congratulations** *At this point the installation is complete and you have a running installation of GNDMS.*

## Testing your installation

The GNDMS contains a client application which tests some basic functionality to ensure your setup is ready to use. In order to run the test-client the following prerequisites must be satisfied:

- You must own a valid grid certificate,
- have access to a grid-ftp-server, which accepts your certificate and offers write permission,
- and of course a running you need a globus container that provides the GNDMS-services, has at least on subspace, and file-transfer enabled.

### About

The test client simulates a standard GNDMS-use-case, it creates as target slice, copies some files into the slice. Then it copies the files back from the slice to some target directory and destroys the slice.

### Setup

For the scenario the following setup is required. On your grid-ftp space create a directory and add some files, e.g. using the following bash command-line:

```
for i in $( seq 1 3 ); do \
    dd if=/dev/urandom bs=1024 count=1024 of=transfer_test$i.dat; done
```

Additionally create a destination directory on the grid-ftp space.

The client reads its properties from a file: `$GNDMS_SOURCE/etc/sliceInOut.properties`. Now it's time to edit this file. All properties whose values contain angle brackets require attention. The file contains comments to every property and hopefully explains itself. When you have finished the file must not contain any angle-brackets, the client will complain if that's not the case.

### Running the test client

Once the setup is complete, load a grid-proxy using:

```
grid-proxy-init
```

Now you can use buildr to fire up the client:

```
gndms-buildr gndms:gndmc:run-test
```

(Or if provided: your grid specific test target) It takes quite some time until the first output appears, be patient. After a successful run your output start with:

```
Connected to GNDMS: Generation N Data Management System VERSION: 0.3-pre "Kylie++"
OK()
Creating slice
```

(of course the version may differ) and end with:

```
Okay, all done. Cleaning up!
    * Destroying Slice
    * Destroying Delegate
Done.
```

Click here to view the full output. If the test runs successfully you should have identical files in your grid-ftp source and destination directory, in that case CONGRATULATIONS!! you have a working GNDMS installation, and can provide data management service for your community.

### Trouble shooting

**The client hangs after the `Copy gsiftp: ...` message.** : This can be a problem with your firewall configuration. It happens when the control-channel can be established but the data-channel is blocked. Please check your firewall setup especially if the `GLOBUS_TCP_PORT_RANGE` environment variable is set correctly and is forwarded by the firewall.

**I'm getting a `GSSException: Defective credential detected` exception.** : This can have to reasons: your certificate-proxy maybe outdated or doesn't exist or your CA directory isn't up to date. In the first case just call `grid-proxy-init` again, in the second refer to the `fetch-crl` section below.

## Advanced Configuration

### Remote Access to container.log

To enable a select group of users to read the container.log from outside, add their DNs to either `/etc/grid-security/gndms-support-stuff` or `$GLOBUS_LOCATION/etc/gndms_shared/gndms-`
Depending on your setup, you need to replace `gndms` with your subgrid name (`ptgrid`, `c3grid`, etc.) in these file names.

To access the log, please load your user credentials (e.g. with `grid-proxy-init`) and run in `$GNDMS_SOURCE`

```
‘env URI="<URI>" ARGS="<ARGS>" gndms-buildr show-log‘
```

where `<URI>` is the EPR of either a DSpace or a GORFX service (see container.log startup section, looks like `https://$HOSTNAME:8443/wsrf/services/gndms/GORFX`) and `<ARGS>` are the arguments that need to be passed to the actual show-log service maintenance call. Please use `env URI="<URI>" ARGS="help"` to obtain a synopsis of possible parameters or leave it empty to retrieve `$GLOBUS_LOCATION/var/container.log` completely.

### Resetting the Database

First, **shutdown the globus container**. Next, in `$GNDMS_SOURCE`, issue

gndms-buildr kill-db

This will delete your database.

### Inspecting the Database

First, **shutdown the globus container**. Next, in `$GNDMS_SOURCE`, issue

gndms-buildr inspect-db

This will open a shell to the derby-ij tool for looking at the internal database of GNDMS.

### Using the Monitor Shell

Please consult the monitor shell guide

## Building GNDMS from Source

Quick Rebuild

A quick full rebuild and reinstallation may be done by executing

```
gndms-buildr rebuild
```

Regeneration of Javadocs

Manually delete `$GLOBUS_LOCATION/doc/api`. Now regenerate the javadocs by executing

```
gndms-buildr apidocs
```

Building Manually from Scratch

```
gndms-buildr clean clean-services # Cleans everything
gndms-buildr artifcats            # Download all 3rd party components
gndms-buildr gndms:model:package  # Compile basic DAO classes
gndms-buildr package-stubs        # Compile service stubs
gndms-buildr gndms:infra:package  # Compile GNDMS framework
globus-stop-container-detached    # Ensure globus is shutdown
gndms-buildr install-deps         # Install dependencies
gndms-buildr package-DSpace       # Compile DSpace service
gndms-buildr deploy-DSpace        # Deploy DSpace
gndms-buildr package-GORFX        # Compile GORFX service
gndms-buildr deploy-GORFX         # Deploy GORFX
globus-start-container-detached   # Restart globus
gndms-buildr gndms:gndmc:package  # Build client
gndms-buildr apidocs              # Build Javadocs (gndms is excluded)
```

**NOTE** *In order to get speedier builds, developers may set* `$GNDMS_DEPS=link`. *This will make* `gndms-buildr install-deps` *symlink dependencies to* `$GLOBUS_LOCATION/lib` *instead of copying them and therefore considerably eases trying out small changes to framework classes. However, when using this method, make sure that required symlinked jar files from* `$HOME/.m2/repository` *and* `$GNDMS_SOURCE/lib`*and* `$GNDMS_SOURCE/extra` *are not deleted accidentally and remain readable for the globus user.*

**NOTE** *Once symlinks have been set up properly, developers may set* `$GNDMS_DEPS=skip` *to skip install-deps alltogether.*

**NOTE** *To even setup symlinks for the service jars, use the* `gndms-buildr link-services` *target.*

Packaging GNDMS

In case you want do distribute your own spin-of GNDMS, we suggest you follow the procedure described below when making a release:

```
cd $GNDMS_SOURCE
vi Buildfile # Set VERSION_NUMBER and optionally VERSION_NAME
gndms-buildr release-build
git commit -m "Made a Release"
git tag gndms-release-ver
git push --tags origin master
find $GNDMS_SOURCE -type f -name '*.class' -exec rm '{}' \;
# Additionally delete */target name/gndms-name and the like
cd ..
mv $GNDMS_SOURCE $GNDMS_SOURCE/../gndms-release-ver
tar zcvf GNDMS-Releases/gndms-release-ver.tgz --exclude .git \
    --exclude *.ipr --exclude *.iml --exclude *.iws \
    --exclude *.DS_Store --exclude *._.DS_Store gndms-release-ver
mv $GNDMS_SOURCE/../gndms-release-ver $GNDMS_SOURCE
```

Now, please upload the tarball and let the world know about it.

Problem Shooting Tips for Development Builds

- Do you need to regen the stubs? `gndms-buildr clean-services package-stubs` to the rescue.
- Symlinks/copies of old jars in `$GLOBUS_LOCATION/lib`.
  find $GLOBUS_LOCATION/lib -type l -name *.jar -exec rm -i {} ;

may help

- If you cant deploy (i.e. globus-start-container balks with one of those 40+-lines stacktraces) it's possible that introduce created an invalid jndi-config.xml which can happen during development but is easy enough to fix: Just make sure there are neither duplicate service nor resourceHome entries in any of the jndi-config.xml files
- This build is not supposed to work on Microsoft Windows
- If you get an error about a missing "test/src" directory simply mkdir -p test/src in the respective services' directory

Other common reasons for a failed container starts are invalid credentials (hostkey/hostcert.pem) or outdated CRLs. In the latter case, the script contained in `$GNDMS_SOURCE/contrib/fetch-crl` may help you. Execute `fetch-crl -o <grid-cert-dir>` with apropriate permissions (Requries `wget`).

# GNDMS Monitor Shell Guide

This is the Monitor Shell Guide for the Generation N Data Management System.

If you have set up your environment as described in the prerequisites section of the GNDMS Installation Guide you may use the GNDMS monitoring and configuration shell to access a running instance of the GNDMS software. This is a little servlet that allows the execution of predefined actions or Groovy 1.6 script code inside the running globus container in order to initialize and configure the database or peek inside the running system for debugging purposes.

On most sites, the GNDMS Monitor Shell is only accessed once to initialize the database during installation.

The GNDMS Monitor Shell is disabled by default and protected by a randomly generated default password. If enabled, it opens a socket on localhost for incoming connections. Please consider that connections are *unencrypted* before configuring it to be accessible from an external network. Again, be aware that you can truely execute arbitrary groovy code with globus user permissions through this channel and therefore be cautious whenever using it. You should enable it only on demand and always disable the service after use.

To enable, edit `$GNDMS_MONI_CONFIG` and set `monitor.enabled` to `true`. Then either restart the container or wait `monitor.configRefreshCycl` ms (defaults to 17 seconds). After this period, the container will load your new configuration and start the monitor shell automatically.

Alternatively, you may set `$GNDMS_MONITOR_ENABLED` to `true` before starting the globus container to enable the monitor.

There are two ways to use the monitor shell, the first allows the execution of predefined actions, while the second runs arbitrary Groovy code.

## Executing Actions

To test the monitoring and configurations shell and retrieve a list of all available actions, execute:

```
moni call help
```

To call an action, execute:

```
moni call <Name of action> <Action Parameters or help>
```

## Executing Groovy Code

This mode of executions is based on http sessions.

`moni init` creates a new session (Default session timeout is 22 mins). `moni open repl foo` to create a new monitor named "foo" in the current session that accepts multiple commands (`repl` is the *run mode* of the monitor. See below for a list of possible run modes).

To use a previously opened monitor, open a second shell and execute:

```
moni send foo $GNDMS_SOURCE/scrips/hello.groovy
```

If you see `Hello, World!` followed by `null` in the first shell you have succesfully enabled the monitor shell.

To close the connection named `foo`, execute `moni close foo`. To destroy your session and close all named connections, execute `moni destroy`. To force the monitor to reread the confuration, execute `moni refresh`. To force a restart even if the configuration has not been altered, execute `moni restart`.

### List of Supported Monitor Shell Run Modes

`SCRIPT` *Default mode* : Accept one send command, do not print result object.

`REPL` : Accept many send commands, always print result objects.

`BATCH` : Accept many send commands, but do not print result objects.

`EVAL_SCRIPT` : Accept one send command, print result object.

**NOTE** Specifying the `<mode>` in `moni open <mode> <connection-name>` is case-insensitive

## Appendix

### Troubleshooting

- If you don't get a connection, check `$GLOBUS_LOCATION/var/container.log` and ensure that the GNDMS Monitor Shell has been started.
- Make sure you have set up your environment as described in the prerequisites section of the GNDMS Installation Guide.
- If you execute moni and nothing happens you might just have forgotten an argument. Currently, moni is just a bunch of helper `bash` scripts that call `curl` and lack proper argument checking. If you do not provide `moni send` with apropriate arguments, it may wait while attempting to read from stdin.

- `monitor.minConnections` should always be >= 2

#### Tips for Script Developers

Inside your own groovy classes, you should always print to `out` or `err` which contain the current monitor's output stream. Plain println only works correctly in the (outmost) script scope or top-level functions.

`out` and `err` properties are added automatically to `Object.metaClass` when a monitor is instantiated. To enable them, `ExpandoMetaClass.enableGlobally()` is called first which affects the semantics of Groovy.

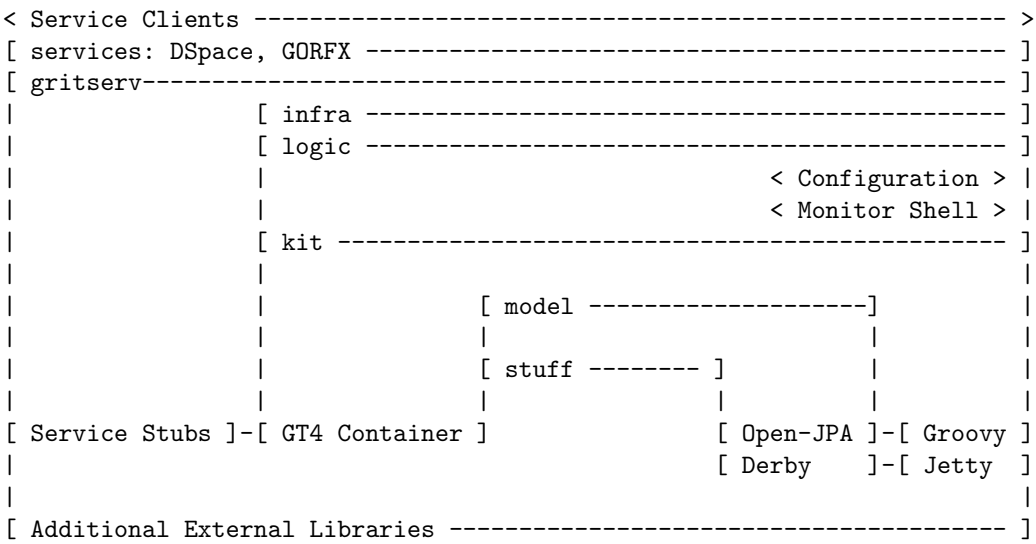
Additional properties like resource homes and singleton resource instances are made available using the same mechanism.

# GNDMS Architecture Primer

This is the GNDMS Architecture Primer. It is developer-level documentation that gives a general overview of the different layers of GNDMS, how they interoperate with Globus Toolkit, as well as important concepts, components and classes. It is incomplete at this point.

## The Layers of the Software Stack

The running GNDMS software stack roughly looks like this:



Legend:

```
[ Module ]        # Software Module
                  # -- modules higher in the Stack
                  # -- depend on modules below them

< Tool   >        # Software Tool
                  # -- what is below is needed to run
```

For the gndmc client, an additional layer is required:

```
< gndmc client ------------------------------------------------------- >
[ gndmc -------------------------------------------------------------- ]
[ services: DSpace, GORFX -------------------------------------------- ]
.....................................................................
```

The software stack for building, installing, deployment and configuration of GNDMS roughly looks like this:

```
< buildr ------------------------------------------------------------- >
[ Build -------------------------------------------------------------- ]
[ Buildr 1.4 --------------------------------------------------------- ]
|             [ introduce ------------ ]                               |
[ JRuby 1.5 ] [ Ant ]                    < Configuration ------------- >
[ Java 2 SDK 1.6 ---------------------- ] < Monitor Shell ------------- >
< local UNIX tools --------------------------------------------------- >
```

## Components

### Component Categories

There are several kinds of components that make up the GNDMS service stack. They can be grouped into various categories

**Main** : GNDMS source code to be found below a top-level diretory and directly built to a jar (Java)

**Service** : Located below `services` and compiled with introduce (XSD, WSDL, Java)

**Client** : Located below `services` and compiled with introduce (Part of service build process, consists of XSD, WSDL, and Java)

**External** : downloaded from the Internet during installation.

**Globus** : part of your local installation of Globus Toolkit 4.

**Build** : All code needed to build and install the software (Ruby and Shell). Of primary relevance are the `Buildfile` in `$GNDMS_SOURCE` and everything in `$GNDMS_SOURCE/buildr`.

**Config** : All code needed to configure GNDMS from the outside (Shell and possibly Groovy)

### Components

**Service Clients** *(Client)* : Any program that accesses GNDMS via WSRF.

**GORFX** *(Service)* : The GORFX service provides the negotiation and execution of data management activites (like Staging and File Transfer) according to a co-scheduling protocol (located below `$GNDMS_SOURCE/services/GORFX`).

**DSpace** *(Service)* : The DSpace service provides management of storage resources (located below `$GNDMS_SOURCE/services/DSpace`).

**gritserv** *(Main)* : Service-level code that is shared between different grid services, like e.g. XSD type conversion (located below `$GNDMS_SOURCE/gritserv`).

**infra** *(Main)* : Main infrastructure code that ties the service code to GNDMS *main* classes. Configuration, Database setup, Dependency injection (located below `$GNDMS_SOURCE/infra`).

**logic** *(Main)* : All 'Business' logic of GNDMS that can be implemented outside of the actual service classes (located below `$GNDMS_SOURCE/logic`).

**Configuration** *(Config)* : Configuration scripts below `$GNDMS_SOURCE` scripts are executing during installation in order to configure a GNDMS instance for its purpose in a given community grid (located below `$GNDMS_SOURCE/scrips`).

**Monitor Shell and Utilities** *(Config)* : The monitor shell (client) allows access to the monitor shell implemented **kit** (located below `$GNDMS_SOURCE/bin`).

**kit** *(Main)* : Utility classes that depend on some functionality from Globus Toolkit and/or **model**. Kit contains the implementation of the GNDMS monitor shell and GridFTP auxiliaries (located below `$GNDMS_SOURCE/kit`).

**model** *(Main)* : Database model classes (located below `$GNDMS_SOURCE/model`)

**stuff** *(Main)* : Various utility classes (located below `$GNDMS_SOURCE/stuff`)

**Service Stubs** *(Service)* : Stub code needed to communicate with the Grid WSRF Services of GNDMS (located inside services).

**GT4 Container** *(Globus)* : The Globus Toolkit 4 WSRF Service Container.

**Open JPA** *(External)* : GNDMS uses Apache OpenJPA 2.0 as its Object Relational Mapper (ORM).

**Derby** *(External)* : GNDMS uses Apache Derby 1.5 as its underlying embedded database.

**Groovy** *(External)* : GNDMS provides support to access the system at runtime by means of executing groovy script code via the monitor shell.

**Jetty** *(External)* : The monitor shell is implemented atop a stripped-down version of jetty.

**Additional External Libraries** *(External)* : GNDMS uses a large selection of 3rd party libraries. Please either consult the `Buildfile` or `$GNDMS_SOURCE/lib/gndms-depencies[.xml]` (post-install) to find out more details. Consult `$GNDMS_SOURCE/doc/licensing` for licensing conditions of 3rd party components.

**Build** *(Build)* : Build scripts are written in ruby and placed in `$GNDMS_SOURCE/Buildfile` and `$GNDMS_SOURCE/buildr/*`.

**Buildr 1.4** *(Build)* : GNDMS relies on Apache Buildr for build, installation, and deployment.

**introduce** *(Build)* : The Introduce Tool from the CAGrid project was used to generate service skeletons below `$GNDMS_SOURCE/services`.

**JRuby 1.5** *(Build)* : **Buildr** needs this.

**Java 2 SDK 1.6** *(All)* : GNDMS has been written in Java.

**Documentation** : Documentation is generated using Javadoc and Jekyll (has been installed in the included JRuby distribution)

GNDMS distribution packages contain a version of JRuby with preinstalled buildr and Jekyll. This is not a part of GNDMS (You could always fallback to your local installation of these tools) and provided for convenience only.

## Suggested Code Walkthrough

- Read the available documentation before entering the code, it will give you a rough idea of how everything is connected
- Get to know the model classes
- Read the action framework (Everything that inherits from de.zib.gndms.logic.action.Action)
- Checkout `infra/src/de/zib/gndms/infra/system/EMTools.java` to understand how actions and the database are connected
- Checkout `Ext*ResourceHome` in **DSpace** to see how resources are persisted.
- Read the `*ServiceImpl` classes to see the actual workflow that is triggered by incoming requests. Follow down to code in **logic** as you see fit.
- Read `infra/src/de/zib/gndms/infra/system/GNDMSystem.java` and
- Read `infra/src/de/zib/gndms/infra/system/GNDMSystemDirectory.java` to understand how GNDMS is bootstrapped and wired

- The monitor is in kit in case you need to touch it

# GNDMS Developer Guide

This is the Developer Guide for the Generation N Data Management System. It is far from complete. It currently contains various tidbits copied together from different Wikis. YMMV. Use the source, luke!

## Writing Webservice Clients

### Setup a Development Environment

- Install GNDMS as described in the installation guide
- Use `gndms-buildr idea` or `gndms-buildr eclipse` to generate template IDEA or eclipse projects.
- You might need to add `$GLOBUS_LOCATION/lib/*.jar`
- Skip `gndms-*.jar`, but
- include `gndms-*-service.jar` and `gndms-*-client.jar`

## Setup a Development Environment for Debugging

- Ensure that the generated modules in your IDE setup compile to the same output path as buildr and that globus, buildr, and your IDE compile using the same JDK.
- Edit your globus scripts such that Java is configured to enable remote debugging and set up a matching run target in your IDE.

**NOTE** *If the globus container is started with* `-debug` *it prints full stacktraces, otherwise not!*

### Writing a Web Service Client

- Take a look at `ProviderStageInClient`
- Do not directly instantiate port types. Always use the associated `PortTypeFooClient` classes to get port type instances.
- If you really need to instantiate port types directly, ensure that the used axis engine is configured with the correct `.wsdd` files. This work is done by `PortTypeFooClient` classes if you use them.

## Notes on Certificate Delegation

To use certificate delegation, two steps are necessary. First, service security settings need to be changed. Second, client and and service code need to be modified slightly to incorporate support for certificate delegation.

### Security Descriptor Basics

**NOTE** This is a very brief description of security descriptors. More advanced configuration is possible, e.g. service method level A&A.

The security descriptor (Short: **SD**) describes authentication and authorization requirements of clients and WSRF web services. The **SD** of a service is configured in the `service` section of the WSDD file.

```
<?xml version="1.0" encoding="UTF-8"?>
<deployment>
    <service>
         <parameter name="securityDescriptor" value="etc/serviceFoo-security-desc.xml" />
    </service>
</deployment>
```

Client security descriptors are loaded directly in the client software:

```
// Client security descriptor file
String CLIENT_DESC = ".../client-security-config.xml";
ClientSecurityDescriptor desc = new ClientSecurityDescriptor( CLIENT_DESC );
//Set descriptor on Stub
( (Stub)port )._setProperty( Constants.CLIENT_DESCRIPTOR, desc )
```

For more details, please consult the documentation on security descriptors.

### Authentication and Authorization

The following example shows how mandatory TLS encryption is enforced with a security descriptor:

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns="http://www.globus.org">
    <auth-method>
        <GSITransport>
            <protection-level>
                <privacy />
```

```
            </protection-level>
        </GSITransport>
    </auth-method>
</securityConfig>
```

This setting must be made both on the server and the client.

For authorization, a gridmap file needs to be set:

```
<authz value="gridmap" />
```

This enables use of the system wide gridmap-file. To use a service specific gridmap file, please add:

```
<gridmap value="etc/gndms_shared/grid-mapfile" />
```

Finally, it is necessary to configure (unless you are using JAAS):

```
<run-as>
   <system-identity />
</run-as>
```

Below is a complete example:

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns="http://www.globus.org">
    <authz value="gridmap" />
    <gridmap value="etc/c3grid_shared/grid-mapfile" />
    <auth-method>
        <GSITransport>
            <protection-level>
                <privacy />
            </protection-level>
        </GSITransport>
    </auth-method>
    <run-as>
        <system-identity />
    </run-as>
</securityConfig>
```

### Client-Side Delegation

This section described delegation from the viewpoint of the client. The client uses the Delegation Service to retrieve the Certificate Chain. This is used to generate a proxy certificate which is sent to the delegation service in order to obtain an EPR for the proxy. This EPR may be passed when accessing resources directly or is sent to factory methods during resource instantiation.

// path to the file containing the proxy cert String proxyFile = ...;

```
// uri of the delegation service
String delUri =  "http://somehost/wsrf/services/DelegationFactoryService"

// port type of our service acquired in the usual fashion
PortType port = ... ;


GlobusCredential credential = new GlobusCredential( proxyFile );

// Create security descriptor for the communication with the delegation service
// This descriptor is not the same we use to communicate with
// the actual service
ClientSecurityDescriptor desc = new ClientSecurityDescriptor();
org.ietf.jgss.GSSCredential gss =
    new org.globus.gsi.gssapi.GlobusGSSCredentialImpl( credential,
   org.ietf.jgss.GSSCredential.INITIATE_AND_ACCEPT );
desc.setGSSCredential( gss );
desc.setGSITransport( (Integer) Constants.SIGNATURE );
Util.registerTransport();
desc.setAuthz( NoAuthorization.getInstance() );

EndpointReferenceType delegEpr =
    AddressingUtils.createEndpointReference( delUri, null );

// acquire cert chain
X509Certificate[] certs =
    DelegationUtil.getCertificateChainRP( delegEpr, desc );

if( certs == null  )
     throw new Exception( "No Certs received" );

// create delegate
int ttl = 600; // a time to life for the proxy in seconds
// the boolean value can be ignored
EndpointReferenceType delegate =
    DelegationUtil.delegate( delUri, credential, certs[0], ttl, true, desc );
```

```
// reuse credentials for this call
( (Stub) port )._setProperty(
    org.globus.axis.gsi.GSIConstants.GSI_CREDENTIALS, gss );

// creates a new resource which uses the delegate, i.e. proxy cert
EndpointReferenceType epr =
    ( (SomePortType) port ).createResource( delegate );
```

### Server-Side Delegation

On the server side, the EPR needs to be used to retrieve the proxy certificate. Additionally, a `DelegationListener` needs to be registered to be informed about proxy state changes (Update, Destroy).

Example service factory method that instantiates a resource:

```
public EndpointReferenceType createResource ( EndpointReferenceType delegate ) {
    SomeResource sr = new SomeResource( );
    sr.setDelegationEPR( delegate );
    ...
    return endPointRefOf( sr );
}
```

The resource needs to be modified accordingly as well:

```
public class SomeResource implements Resource {

    SomeResourceHome home;
    GlobusCredential credential;

    public void refreshRegistration( final boolean forceRefresh ) {
        // do refreshing stuff if required
    }


    public void setCredential( final GlobusCredential cred ) {
        credential = cred;
    }


    public GlobusCredential getCredential( ) {
        return credential;
    }
```

```
    public void setDelegateEPR( final EndpointReferenceType epr ) {

        SomeDelegationListener list =
        new SomeDelegationListener( getResourceKey(), home );
        try {
            // registers listener with the delegation service
            DelegationUtil.registerDelegationListener( epr, list );
        } catch ( DelegationException e ) {
            e.printStackTrace();
        }
    }

    // other service specific stuff here ...
 }
```

The container will be calling `get/setCredential` on the listener interface. A simple default implementation follows:

```
public class SomeDelegationListener implements DelegationListener {

    private static Logger logger = Logger.getLogger( SomeDelegationListener.class );
    private String regristrationId;
    private ResourceKey resourceKey;
    private ResourceHome home;


    public SomeDelegationListener() {
    }


    public SomeDelegationListener( final ResourceKey resourceKey,
final ResourceHome home ) {
        this.resourceKey = resourceKey;
        this.home = home;
    }


    public void setCredential( final GlobusCredential credential )
throws DelegationException {

         try {
           SomeCredibleResource res =
           ( SomeCredibleResource ) home.find( resourceKey );
           res.setCredential( credential );
```

```
        } catch ( ResourceException e ) {
          logger.error( e );
        }
    }


    public void credentialDeleted() {
       // Can notify the resource
    }

    // getters and setters for the instance vars are omitted for the sake of shortness
    // ....
}
```

The `setCredential` method will be called at listener registration time.

With these extensions, a resource has access to the credentials of the user to which the proxy belongs.

### Using Delegation with Proxy Certificates

Service Orchestration

The main purpose of certificate delegation is to allow a service to call another service on behalf of the user. Let's assume `SomeService` is a client of `AnotherService`. In the following example `AnotherService` is called by `SomeService` with the proxy credentials by first loading them into the `ClientDescriptor`:

```
AnotherPortType port = ...;
    ( (Stub) port )._setProperty( org.globus.wsrf.security.Constants.GSI_TRANSPORT,
                        org.globus.wsrf.security.Constants.ENCRYPTION );
            // SIGNATUR should also work
    org.ietf.jgss.GSSCredential gss =
    new org.globus.gsi.gssapi.GlobusGSSCredentialImpl( credential,
            org.ietf.jgss.GSSCredential.INITIATE_AND_ACCEPT );
    ( (Stub) port )._setProperty(
    org.globus.axis.gsi.GSIConstants.GSI_CREDENTIALS, gss );
    ( (Stub) port ).doAnotherThing();
```

Now, in `AnotherService`, the caller DN (`null` in anonymous communication) is obtainable by calling:

```
org.globus.wsrf.security.SecurityManager.getManager().getCaller();
```

This may be mapped to local UNIX users via the grid-map mechanism:

```
org.globus.wsrf.security.SecurityManager.getManager( ).getLocalUsernames()
```

Export Proxy Credentials to a File

```
public void storeCredential( Sting filename ) {
     try {
         File f = new File( filename );
         FileOutputStream fos = new FileOutputStream( f );
         GlobusGSSCredentialImpl crd =
         new GlobusGSSCredentialImpl( credential, GSSCredential.ACCEPT_ONLY );
         fos.write( crd.export( ExtendedGSSCredential.IMPEXP_OPAQUE  ) );
         fos.close();
     } catch( Exception e ) {
         // an exception --- do something
     }
}
```

The resulting file is structured as follows:

- Proxy certificate generated last
- Private key of this certificate
- Certificate chain in descending order

The exported proxy may be verified manually with openssl by first splitting this file into the `head` (containing everything but the certificate chain) and the `tail` (containing the certificate chain) and setting `$OPENSSL_ALLOW_PROXY_CERTS=1`. Now execute:

```
openssl verify -CApath /etc/grid-security/certificates -CAfile tail head
```

If everything is ok, `openssl` will print

```
head: OK
```

Otherwise a lengthy error message will be shown.

Another way to acompish this is to install the tool grid-proxy-verify, which handles the proxy-file without the need of splitting. A look at the source code is an interesting read concerning the details of proxy verification with openssl.

## Contract Semantics

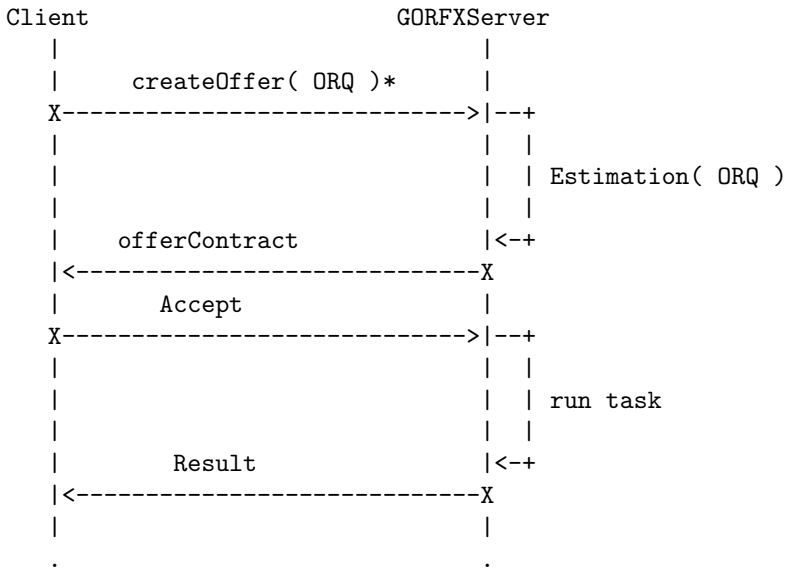
This section describes the precise semantics of offer contracts in *GNDMS*. An *offer* is an negotiable offer for the execution of a data management task (like Staging, Transfer, and Publishing). Offer contracts may specify requirements on execution time, duration and location. Offers are negotiated between a client and server in rounds until agreement is reached and a contract is succesfully established.

Client and server roles are taken by different participants. For example, a grid meta scheduler may be a client to a central data management site that runs GNDMS, while the same site may be a client to a data provider site in a different negotiation.

### Protocol

The protocol consists of three steps.

- Client send an OfferRequest with desired task and requirements
- Server replies with an offer contract that tries to match clients requirements.
- Client either accepts the offer. In this case, the protocol is finished with the creation of a task resource that allows the client to monitor task execution and to fetch results. Otherwise, the client is free to discard the offer and redo the protocol with another site or another contract.



### Contract Structure

A contract consists of

- An (optional) point in time called `IfDecisionBefore`, **IDB** for short
- An (optional) point in time called `ExecutionLikelyUntil`, **ELU** for short
- A point in time or an offet calles `ResultValidUntil`, **RVU** for short, or **Delta-RVU**, respectively
- An (optional) size estimation calles `EstMaxSize`, **EMS** for short (upper bound on the number of bytes of result data)
- An (optional) set of key-value apires called `RequestInfo`, **RI** for short. **RI** may be used to pass additional information like remarks, warnings etc. From a middleware point of view, **RI** is *not* part of the contract.

### Contract Semantics

**Precise contract semantics** : If the Offer is accepted before **IDB**, the task will be executed before **ELU** with high probability. Results are made available until **RVU** as long as they do not need more than **EMS** bytes of storage.

If **IDB** is missing, it is interpreted as an *arbitrary, undefined point in the future*. This is currently not supported by the software but specifiable according to the underlying XSD schema.

If **ELU** is missing, it is interpreted as *arbitrary, unknown task execution duration*. This is currently not supported by the software but specifiable according to the underlying XSD schema.

### Generic Client Restrictions

**IDB** is mandatory. All mandatory invariants need to be fulfilled.

### C3-Grid Data Provider Server Restrictions

Contract semantic variables are mapped to staging properties as detailed below:

```
IDB = c3grid.StageFileRequest.Estimate.IfDecisionBefore,
ELU = c3grid.StageFileRequest.Estimate.ExecutionLikelyUntil,
RVU = c3grid.StageFileRequest.Estimate.ResultValidUntil,
EMS = c3grid.StageFileRequest.Estimate.MaxSize,
RI  = c3grid.StageFileRequest.Estimate.RequestInfo
```

- Data providers must specify an **ELU** that must be identical to the **ELU** requested from the client
- **IDB** and **RVU** may not be modified by the server. In the case of **RVU** < **ELU**, the client should discard the request.
- Client **EMS** may be discarded or overwritten by the server in his offer
- **RI** is filtered for keys

**Summary** : For staging requests to data providers, it is sufficient to specify **ELU** in ms and **EMS** in bytes, and to optionally include key-value data in **RI**

#### Support for Missing Timing Estimates

#### Contract Invariants

Depending on how the contract requested by the client looks like, some invariants need to be fulfilled:

**ELU, RVU** requested : **IDB** < **ELU** and **IDB** < **RVU** (Therefore in practice, choose **IDB** < **ELU** < **RVU**)

**Delta-ELU, RVU** requested : **IDB** < **RVU** (Therefore in practice, choose **IDB** + **Delta-ELU** < **RVU**)

**ELU, Delta-RVU** requested : **IDB** < **ELU** (and **RVU** is **ELU** + **Delta-RVU** and therefore **ELU** <= **RVU** always holds)

**Delta-ELU, Delta-RVU** requested : No invariants, it holds that start time **ST** <= **IDB**, completion time **CT** = **ELU** = **ST** + **Delta-ELU**, **RVU** = **CT** + **Delta-RVU** and therefore always **ELU** <= **RVU**

It always holds that **ST**<=**IDB**<=**CT**<=**ELU**. If **Delta-RVU** was requested. Additionally always **CT** <= **ELU** <= **RVU** is true.

Clients need to honor all invariants. Servers need to honor all invariants which do not contain **RVU**. Servers may only modify **ELU**.

**Contract** : GNDMS negotiates *contracts* with clients about task execution. A contract specifies what is to be done, and optionally when and where it is to be done by GNDMS on behalf of the client. *Contracts* are accepted *Offers*.

**Data Provider** : In C3-Grid, data providers are sites that run GNDMS with the Staging Plugin in order to grant access to their local climate data archives.

**DMS** : Often used for the (or a) central data management coordination site of a community grid.

**DMS-Publish** : In C3-Grid, load balancing publish to a dedicated storage server.

**DMS-Staging** : In C3-Grid, indirection of a staging request to a matching data provider or cache.

**DSpace** : Workspace management service of GNDMS. Each DSpace is structured into a set of *subspaces* (Logical stroage group). Each subspace consists of multiple *slices* (Non-hierachical container of files).

**GNDMS** : Generation N Data Management System. A data management solution for community grids based on the Globus Toolkit 4 Middleware.

**GORFX** (aka **G**eneric **O**ffer **R**equest **F**actory **X**) : Service for the negotation of data management task execution.

**Offer** : Cf. *Contract*, offers are the subjects of contract negotiation, Offers are not-yet accepted contracts.

**Offer Request** : Description of a task and required *offer* constraints.

**Publish** : In C3-Grid, publishing of intermediary results

**Publish-Host** : Host that provides storage resources for *Publish*. Needs to run *DSpace* and *GORFX* configured for support of the *Publish* task.

**Staging** : In C3-Grid, import of climate data from external archives into the data management infrastructure of the community grid.

## License

GNDMS is distributed under the Apache License 2.0

Please consult the file LICENSE and everything below 'doc/licensing' in your release tarball for details.

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that

is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and

You must cause any modified files to carry prominent notices stating that You changed the files; and

You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the

following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

Accepting Warranty or Additional Liability. While redistributing the Work

or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS