
OMII-UK Commissioned Software¹

Product Deliverable and Integration Specification

Revision: 0.80

Authors:

Gary Li
Stephen Crouch
Steven Newhouse
Simon Hettrick
Tim Parkinson
Justin Bradley

Open Middleware Infrastructure Institute
Room 6005, Building 21 (Faraday)
Highfield Campus
University of Southampton
Hampshire, SO17 1BJ
Tel: +44 (0)2380 58787

¹ Commissioned Software used to be called Managed Programme in the first phase of the OMII-UK project. The term MP for short may appear as a reference in the code or documents.

Contents

<i>1</i>	<i>Introduction and Overview</i>	<i>4</i>
1.1	Process and Activities.....	4
1.2	Definitions and Relations.....	5
<i>2</i>	<i>Source and Binary Distribution Structure</i>	<i>6</i>
2.1	Overall Distribution Directory Structure.....	6
2.2	Source Distribution Structure and Contents.....	7
2.2.1	LICENCE	7
2.2.2	README.txt	7
2.2.3	ReleaseNotes.txt	7
2.2.4	build.xml	8
2.2.5	src Directory	8
2.2.6	lib Directory	8
2.2.7	licenses Directory	8
2.2.8	dist Directory	8
2.3	Binary Distribution Structure and Contents.....	8
2.3.1	LICENCE	9
2.3.2	manifest.properties	9
2.3.3	build.xml	9
2.3.4	README.txt	9
2.3.5	doc Directory	9
2.3.6	bin Directory	10
2.3.7	Lib Directory	10
2.3.8	config Directory	10
<i>3</i>	<i>Build Process</i>	<i>10</i>
3.1	Required Ant Targets.....	10
3.2	Shared Resources.....	10
<i>4</i>	<i>Installation Process</i>	<i>11</i>
4.1	Overview of the Process.....	11
4.2	Install-time Ant Properties.....	11
4.3	Required Ant Targets.....	12
4.4	The Ant Targets Invocation Order.....	13
4.5	Server Web Service Deployment Methods.....	13
4.6	manifest.properties.....	14
4.7	Database Usage.....	14
4.7.1	Table Namespace in an existing database	14
4.7.2	Database Namespace	15
<i>5</i>	<i>Documentation requirements</i>	<i>15</i>
<i>6</i>	<i>Release check list</i>	<i>18</i>
6.1	Code check - Building the binary distributions from source.....	18
6.1.1	Verification Using the Automated Build Software	18

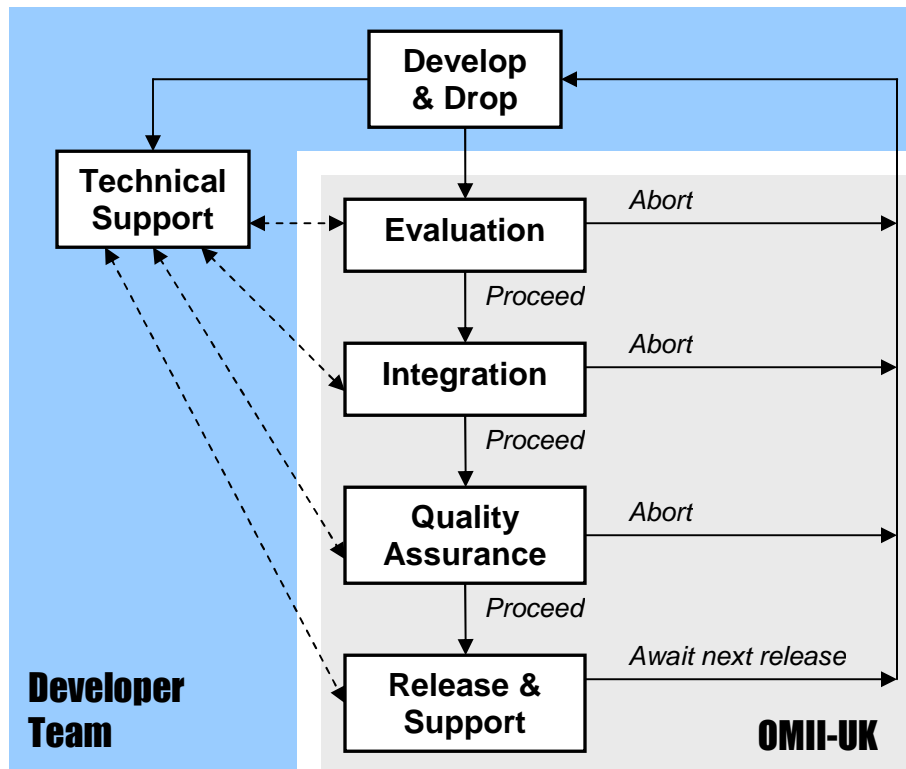
6.1.2	Manual Build Verification	18
6.1.3	Verifying the Server Binary via Installation	18
6.1.4	Verifying the Client Binary via Installation	19
6.2	Document check.....	19
6.3	Packaging check.....	19

1 Introduction and Overview

This document defines a standard interface by which commissioned software from our providers can be integrated into OMII-UK software releases. The integration point will be through 'Ant' build scripts, provided by the software provider. These are called by the OMII-UK build scripts to build binaries from the source distribution and then the build scripts in the binary distribution can be called again by the OMII-UK installer to install the software onto a machine. We specify how we expect the software package source to be delivered to us, how we can then build a client and server distribution, and how we can install the client and server distribution onto a machine.

1.1 Process and Activities

A commissioned software release coming in from an OMII-UK software provider will go through the following process:



In summary:

1. **Develop & Drop:** In this step the software provider will develop, test and package up the source distribution according to OMII-UK requirements and follow the required procedure to release a source package to OMII Southampton.
 2. **Evaluation:** OMII Southampton will evaluate a software package to determine the function, usage and quality of the product. To what extent does it integrate successfully with the OMII-UK release? What documentation is provided and how good/useful/complete is it? Any outstanding issues that are identified will be passed back to the developer team.
 3. **Integration:** The release is integrated into the OMII-UK release.
 4. **Testing & QA:** OMII-UK will carry out an internal, integration-independent QA process, to more thoroughly ascertain the quality of the product and identify any further issues that can be passed back to the developer team.
 5. **Release and Support:** OMII-UK will release commissioned software for its community with all supporting material in place and carry out support and training as an ongoing activity.
-

After develop and drop, the developer team adopts a technical support role in the following phases:

- **OMII-UK ingest support:** provided during the evaluation, integration and quality assurance process.
- **Post-release support:** user-level support after the release.

During the development process, an issue (or issues) may be raised to which there does not appear to be an adequate resolution. If such an issue occurs during the evaluation, integration or testing and QA steps, there is the opportunity to abort the process and go back to step 1.

1.2 Definitions and Relations

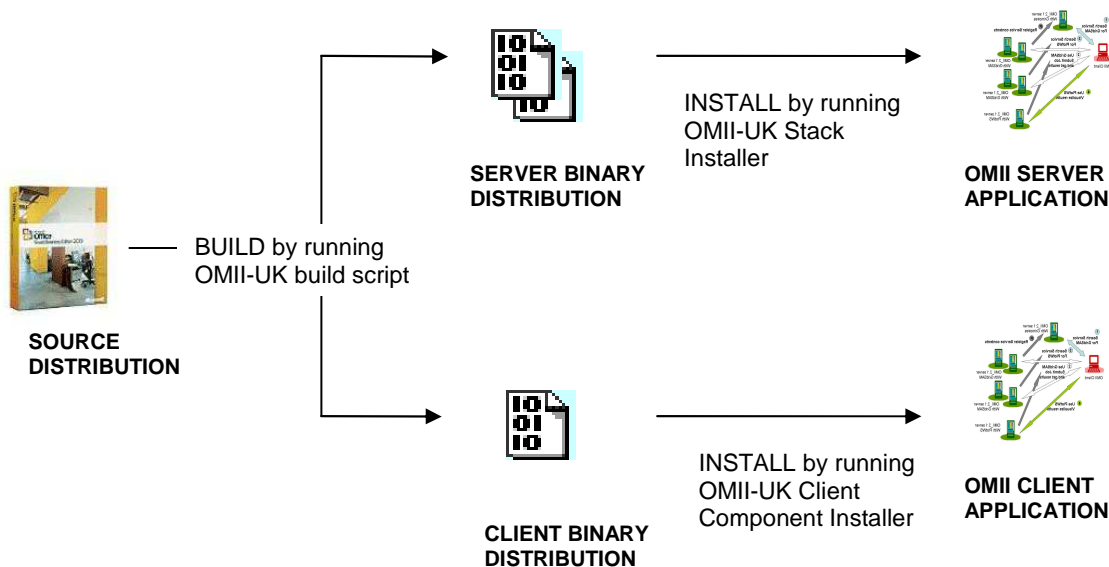
We define the following three software states:

1. **Source Distribution:** This is a directory structure provided to OMII-UK by the developer team that contains all the pre-built software and documentation related to that particular software release. It is in .tar.gz format.
2. **Binary Distribution:** These (client and/or server) releases are generated from the source distribution. The server release is in .tgz format the client release is in .zip format.
3. **Installed Component:** The software is installed into an OMII-UK software installation which is ready to be used and is in working order. At this point the software becomes an installed *component*.

The following two actions facilitate the transition to the next of the above states:

1. **Build (conducted at OMII-UK):** This process generates the client and/or server binary distributions from a source release by running the *OMII-UK build script*. It calls an Ant build-all target in the source distribution's build.xml file to perform the generation. Once this is conducted the resultant distributions can be installed by the OMII-UK software installers.
2. **Install (conducted by the individual user):** An action undertaken by either the OMII-UK client or OMII-UK server installers to install one of the binary distributions to the OMII-UK Client or OMII-UK Server respectively.

The following illustrates the above:

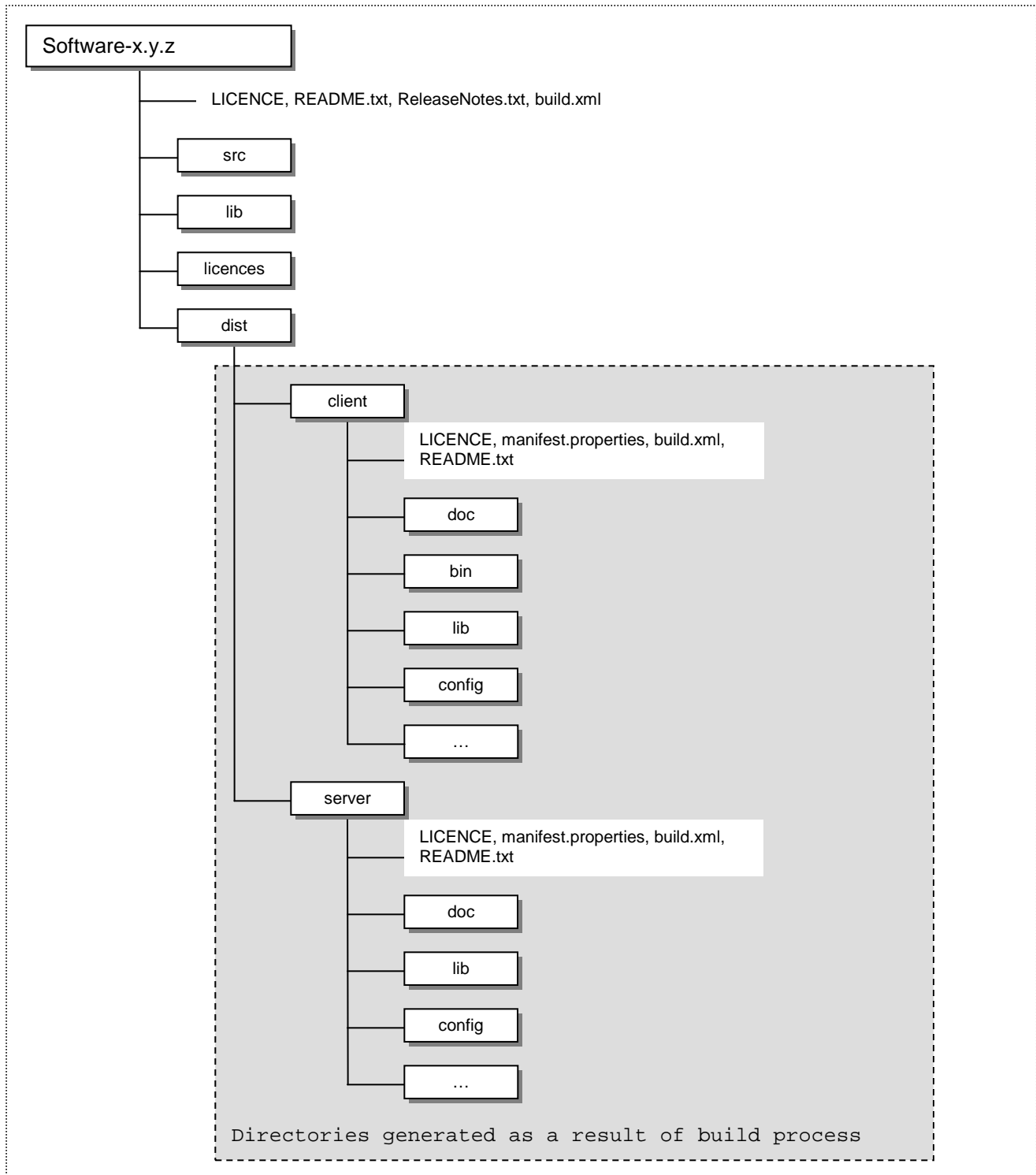


Whilst a source distribution can generate both client and server binaries, either one is optional (although at least one of these must exist).

2 Source and Binary Distribution Structure

OMII-UK expects that any commissioned software should follow the following package requirements.

2.1 Overall Distribution Directory Structure



Looking at the above structural view, the elements in the shaded area are generated from the build process. We will define the detailed requirements and structure for the source and binary distributions in the following sections.

2.2 Source Distribution Structure and Contents

The source distribution should be delivered with the following directory structure:



Referring to the above diagram, a software package named SOFTWARE must be released to OMII-UK with a name <SOFTWARE-x.y.z> in tar.gz format. Therefore the name will be:

<software>-<x.y.z>.tar.gz

Where software is a legal string with no illegal ASCII code in it and should not contain any hyphens. x.y.z is the version of the software. No two releases of the software can use the same release version. Major stable releases will be denoted by an 'even' minor release number, e.g. 1.0.0, 1.2.0. Updates to stable distributions will be released as required (e.g. bug fixes, security patches, etc.) and will be denoted by an increment in the point release i.e. 1.0.0 to 1.0.1. These releases will at a minimum have sufficient testing to verify that the fixes are successful and have not affected established capability. In between stable releases there may be intermediate experimental developer releases, which are denoted by an 'odd' release number, e.g. 1.1.0.

2.2.1 LICENCE

Main product licence in ASCII text file format covering the software in this release.

NB: This should be spelt LICENCE as opposed to LICENSE.

2.2.2 README.txt

ASCII text format file that contains general information about the software. e.g. as software name, version, authors, packaging date and structure, and contact details. It needs to include anything that a reader needs to know in particular for generating binaries from this source release e.g. any specific requirements.

2.2.3 ReleaseNotes.txt

To avoid possible integration problems upon a release, each release should include concise answers to a series of short questions. The questionnaire should be answerable in 15 minutes:

1. What environment has been used to test the software for correct operation? Include details such as operating system, java version, the version of the OMII-UK software to which the binaries were installed, etc.

-
2. What instructions should be followed to verify the correct operation of the release?
 3. To what extent have the identified issues in any previous releases been addressed? A list of bug numbers and short resolutions would be ideal.
 4. Are there any deviations from the Commissioned Software Integration Specification? If so, please list them.
 5. To what extent do the binary distribution(s) integrate with the OMII-UK security model (i.e. WSS4J)?

2.2.4 build.xml

The Ant script invoked by the build process that builds the client and/or server binary distributions from the included source (see section 3).

2.2.5 src Directory

Sources required to build the binary distributions. This directory must include all source code and documentation files (e.g. LaTeX) which are processed to generate the 'real', used documentation in the binaries.

2.2.6 lib Directory

Contains any third party resources (jars, packages, etc.) required by the project that can be redistributed with the package. It is the responsibility of the developer team to ensure that any packaged third party components can be redistributed without any licensing and/or legal issues.

An optional EXTERNAL.txt file may reside here to outline any other resources required that must be obtained elsewhere that cannot be redistributed.

2.2.7 licenses Directory

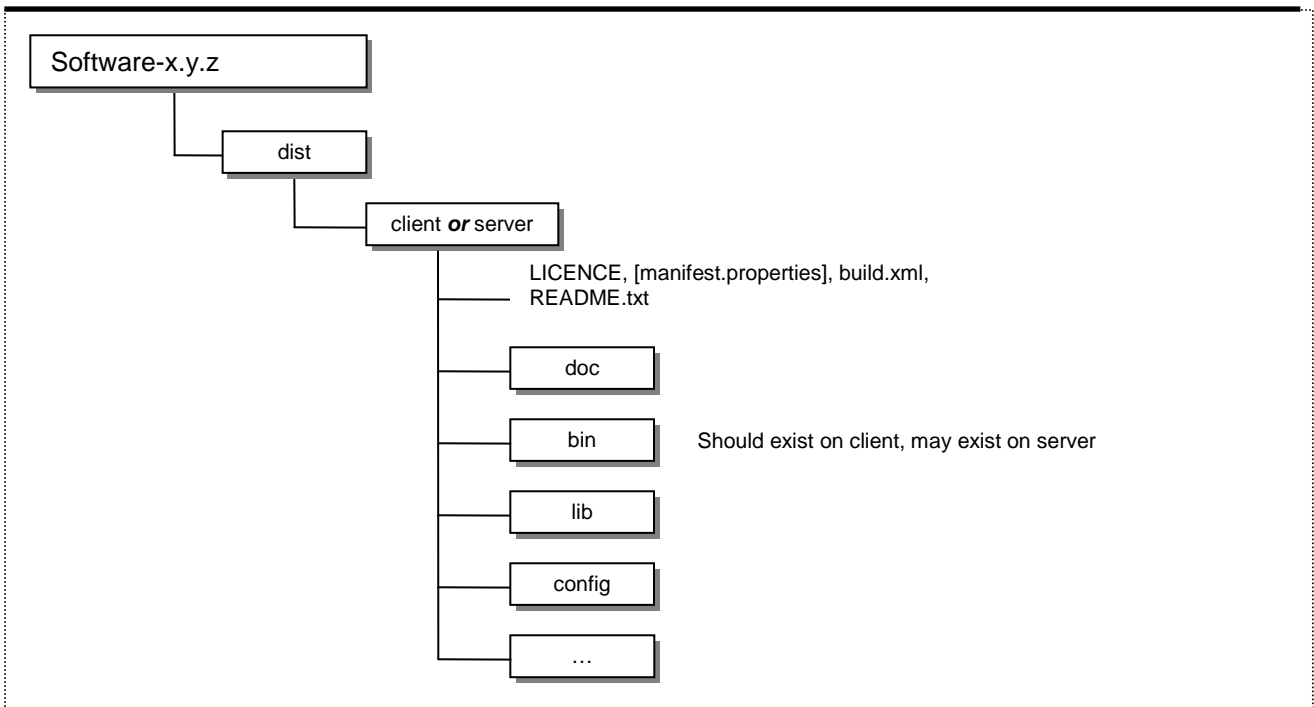
License files (with manifest) for the third party resources contained in the lib directory.

2.2.8 dist Directory

When the build process is complete, the client and/or server directories will be created, containing the client and/or server sides of the product respectively. Each of these directories will be automatically packaged into distribution archives by the OMII-UK build process.

2.3 Binary Distribution Structure and Contents

A client or server binary distribution, generated from the source distribution, should be created with the following directory structure:



It is not necessary to archive the directories into .zip or .tgz formats; this is accomplished automatically by the build process itself.

2.3.1 LICENCE

Main product licence in ASCII text file format covering the software in this release. If this file exists, it will be displayed automatically for acceptance by the OMII-UK server/client installers prior to installation of the component. It must be accepted by the user in order for the installation to proceed.

2.3.2 manifest.properties

The file that contains a list of properties that outline various configuration properties of the installation (see section 4.6). This file is generated *during* installation, but is included at this location since this is where the installer will expect to find it (if generated) following installation.

2.3.3 build.xml

The Ant script invoked by the installer that installs the client and/or server binary distribution itself (see section 4).

2.3.4 README.txt

ASCII text format file that contains general information about the software. e.g. software name, version, authors, packaging date and structure, and contact details. In the case of a binary distribution it needs to include anything that a reader needs to know for installation e.g. any specific requirements.

2.3.5 doc Directory

To contain any rendered ready-to-use documentation for product users. See section 5 for more details on the documentation requirements.

2.3.6 bin Directory

For a client, this may include any scripts/programs used to invoke the functionality of the product. Optionally for a server, this may contain various administrative programs for testing/configuring the server, for example.

2.3.7 Lib Directory

Contains any jars required at run-time by the software. Please make sure that any jars that exist in the OMII-UK installation as a shared resource (see section 3.2) are not duplicated in the installation of the product.

2.3.8 config Directory

Contains any configuration resources required by the software e.g. properties files.

3 Build Process

The OMII-UK build process, which is Linux specific, uses Ant to invoke expected targets in a source distribution's build.xml to generate the server and/or client binaries. This section should be read in conjunction with section 2 which covers the structure of the source distribution.

The build process is an offline activity that is performed at OMII-UK and not by users.

3.1 Required Ant Targets

The OMII-UK build process runs the OMII-UK build script on the software source distribution.

We expect the following build targets to be supported by the top level build.xml:

- `build-all`: Build the client, server and documentation elements of the distribution in the appropriate directory (effectively call the `build-server`, `build-client`, `build-doc` targets).
- `build-server`: Build the server distribution
- `build-client`: Build the client distribution
- `build-doc`: Build dynamically generated documentation, e.g. Javadoc, compile Latex files, etc. and populate client and server documentation distribution trees.
- `test`: Run unit tests, or other tests that are not dependent on consuming live services.
- `clean`: Remove all the intermediary build artefacts

The build process invokes the `build-all` target, followed optionally by the `test` and `clean` targets.

It can be assumed that there is one environment variable defined at build time (`$OMII_HOME`) that can be utilised in the Ant build.xml (e.g. `${env.OMII_HOME}`) which points to an installed OMII-UK Server. This can be used to locate a shared lib set to compile the source distribution against when necessary (e.g. adding `${env.OMII_HOME}/jakarta-tomcat-5.0.25/shared/lib` to the compilation classpath).

3.2 Shared Resources

To promote more space-efficient distributions, both the OMII-UK client and OMII-UK server provide shared library and classes directories where commonly used resources are stored to be utilised by all products. These should be considered read-only resources.

Where possible, to assure compatibility, it is recommended that a source distribution compiles against this common set of resources (as well as any additional resources that are required by the product), and installed binaries use these resources at run-time. Any jars that already exist within the client/server shared OMII-UK resources directories resident should be removed from your package. It is recommended that the software provider installs the latest OMII-UK software release, and use the client (`OMIIClient/conf` and

OMIICLIENT/lib) resources and server (*OMII/jakarta-tomcat-5.0.25/shared/lib* and *OMII/jakarta-tomcat-5.0.25/shared/classes*) resources for development purposes.

4 Installation Process

4.1 Overview of the Process

To maintain a unique structure in each OMII-UK installation, the software will be installed into a pre-determined location with a particular configuration, the details of which are provided to the binary in question. Each software binary build.xml has to be parameterised with this overall requirement, and the OMII-UK installer will pass a set of Ant properties to the binary's installer at install time to facilitate this.

The OMII-UK installation scripts will perform the following tasks in the following order:

1. Determines the currently installed components (examines the *OMII_HOME/records* directory). Loads in manifest.properties files for each component that contain individual installed component information (see section 4.6 for details on manifest.properties files).
2. Displays the currently installed components to the user, and a list of components they are able to install from the software release (i.e. that are not installed). The user can then select those components that they wish to install.
3. The user must then accept any licences that exist in each distribution (LICENCE file located in *<projectand-version>/dist/server/LICENCE*). If they decline a component licence, that component and any dependent components are not installed.
4. Determines dependency problems by examining installed components and installable components. It relays any missing, required component information to the user. Thus, the user can install components based on this information by reselecting those components and accepting any licences they had previously rejected. This cycle continues until dependency problems are resolved or the user quits.
5. Determines component install order.
6. Installs components based on this install order.
7. Creates a new entry in the *OMII_HOME/records* directory for each newly installed component (effectively a copy of the manifest.properties file for that component including information as to whether a licence was accepted).

In addition, throughout the installation process, each commissioned software component installer is passed the properties from all components installed up to that point (i.e. all properties files in *OMII_HOME/records*). Each commissioned software component property set has its PROJECTNAME as a prefix (e.g. MyProject.name, MyProject.version). This enables each software component access to properties of previously installed software components, either during this installation period or any previous installation period. This is in case a component requires configuration information during its installation for another commissioned software installed component upon which it is dependent. Finally, this information can be used for problem diagnosis and troubleshooting.

Please be aware that you can make use of this file for your own installation and configuration purposes too. The creation of a lot of separate build.properties and configure.properties files is discouraged.

4.2 Install-time Ant Properties

At install time a set of properties are defined that must be utilised by the client/server binary.

For the client, only one property, *omii.client.home*, is defined that holds the location of the installed OMII-UK Client to which the client binary will be installed. For the server, the following properties are defined:

OMII-UK Installed directories:

- *omii.server.home*: Installation root directory for the server.
- *tomcat.home*: Installation root directory for the Tomcat distribution.

OMII-UK Container hostname and port:

- `tomcat.port`: Port on which the tomcat installation is running.
- `tomcat.host`: Fully qualified hostname of the tomcat installation machine.

OMII-UK Ownership and permissions:

- `tomcat.ownership.owner`: OMII's Tomcat owner (e.g. `omii_tomcat_owner`) – root install only.
- `tomcat.ownership.group`: OMII's Tomcat group (e.g. `omii_tomcat`) – root install only.
- `tomcat.setup.users`: Whether to change the ownership and group of the installed webapp to above `owner:group` – 'on' = yes.
- `tomcat.permissions.files`: The permissions that should be applied recursively to directories within the webapp directory, e.g. 660 or o-rwx.
- `tomcat.permissions.directories`: The permissions that should be applied recursively to files within the webapp directory, e.g. 750 or o-rwx.

OMII-UK Database usage:

- `omii.mp.database.connectionmanager`: The class name from which a new instance of the connection manager can be instantiated.
- `omii.database.hostname`: The IP address that omii DBMS is installed on.
- `omii.database.port`: The port number that is used to connect to the omii DBMS.
- `omii.mp.database.url`: The JDBC URL that is used to connect to the database instance.
- `omii.mp.database.username`: The username required to authenticate to the database instance.
- `omii.mp.database.password`: The password required to authenticate to the database instance.
- `omii.mp.database.driver.location`: the location where the database driver is located. You can include it into your classpath.

OMII-UK admin:

- `tomcat.administration.username`: Tomcat's administration account username, if required for hot deployment in the 'deploy' target using Axis' AdminClient.
- `tomcat.administration.password`: Tomcat's administration account password, if required for hot deployment in the 'deploy' target using Axis' AdminClient.

4.3 Required Ant Targets

The OMII-UK build installation process runs the OMII-UK installers (client or server) on a binary distribution.

We expect the following build targets to be supported by an installation build.xml:

- `release`: A one line string, echoed to the standard output, that encapsulates the project, this release, and any other relevant information, e.g. 'CoolGridService v1.0.1 April 2005 <http://www.xyz.org/cgs>'
 - `verify-install-env`: Check that the environment will support the install and deploy targets.
 - `verify-uninstall-env`: Check that the environment will support the uninstall and undeploy targets.
 - `pre-install`: Prepare the required classes and jars in the relevant distribution directory for installation. This may include unpacking a war file and altering the configuration files contained within it for the local environment before rebuilding the war in preparation for deployment.
 - `post-install`: The Tomcat server will not be running during the invocation of this target. This allows permissions in the webapp to be changed without side-effects within Tomcat.
 - `uninstall`: Remove files or directories from the installation directory that require Tomcat to be running but do not cause side-effects within Tomcat.
 - `deploy`: Deploy the service into the hosting environment using the project provided WSDD file and/or the Tomcat admin tool.
 - `undeploy`: Undeploy the service from the hosting environment using the project provided WSDD file and the Tomcat admin tool.
-

-
- `test-install`: Run tests against the running service to verify that it is working correctly.
 - `clean-install`: The Tomcat server will not be running during the invocation of this target. This allows the deletion of files or directories that may have remained in the install directory (or even the entire `webapps` directory itself) without side-effects within Tomcat.
 - `configure-db`: Create the relevant database tables using the namespace defined in section 4.7.
 - `clean-db`: Drop the relevant database tables created to support this service.

4.4 The Ant Targets Invocation Order

The OMII-UK installers will invoke the Ant build script targets in the following order to install the software:

1. *tomcat running*
2. `verify-install-env`: On failure no further targets will be called.
3. `pre-install`: On failure call `uninstall`.
4. `configure-db`: On failure call `uninstall`. *Only invoked for a server installation.*
5. `deploy`: On failure call `undeploy` and `uninstall`.
6. *stop Tomcat*
7. `post-install`: On failure call `undeploy` and `uninstall`
8. *start Tomcat*
9. `test-install`: On failure exit the OMII-UK installer.

The OMII-UK uninstall scripts will call the build script targets in the following order on a running tomcat instance to uninstall the software:

1. *tomcat running*
2. `verify-uninstall-env`: On failure no further targets will be called.
3. `undeploy`
4. `clean-db`: *Only invoked for a server uninstallation.*
5. `uninstall`
6. *stop Tomcat*
7. `clean-install`
8. *start Tomcat*

For the server binary distributions, it should be stressed that it is not the responsibility of the install scripts to control the starting and stopping of the OMII-UK Tomcat container. Those actions are conducted by the OMII-UK server installer itself (indicated by action and state in italics above).

4.5 Server Web Service Deployment Methods

There are a number of ways that a server's install build.xml can deploy a web service to the OMII-UK Tomcat container. The two most common (mutually exclusive) ways are:

- **Offline deployment:** Whilst the Tomcat server is shutdown, the *post-install* target is invoked. The component's install script creates the `webapps/<project-name>` directory and copies the webapp files into that directory. As part of this target, it also:
 - **If root:** Sets the ownership of the directory and contents to the OMII Tomcat container owner and group passed via the `omii.ownership.owner` and `omii.ownership.group` properties respectively.
 - **If root or non-root:** Sets the permissions of the directory and contents to the permissions denoted by the `tomcat.permissions` property.
- **Hot deployment:** Whilst the Tomcat server is running, the *deploy* target is invoked. The components' install script deploys a webapp .war file into the container using Axis' AdminClient, using the username and password

In both cases, the *pre-install* target is invoked whilst Tomcat is running, before either of the above operations. The component install script can perform any necessary preparation for either of these events (e.g. unpacking archives, configuring files, preparing the .war, etc).

4.6 *manifest.properties*

There are two manifest.properties files that are required in both binary releases. They can be created using any method with which one is comfortable with (e.g. as a side effect of the component installer or statically, whichever is appropriate). However, the files must be readable by the Java property package. The *propertyfile* optional Ant task should not be used for this purpose, because it causes escaped characters to be inserted into the properties file. The contents of the file, apart from the following mandated properties, can be anything that you feel will be useful, in particular for other software packages that may be dependent on yours. We require the following mandated information:

1. **Global static information about the product:** It should serve the purpose of quickly and accurately describing the distribution.
 - Official or well-known name: `name=<PROJECTNAME>` e.g. `APackage`
 - Version of the product: `version=<X.Y.Z>` e.g. `1.0.0`
 - Other optional information such as maker, contact, url, etc..
2. **Dependencies:** A list of other dependent packages that is required for operation. The purpose of this information is for integration and cooperation with other components. With this in mind, you can add any other additional properties deemed necessary or useful that may be used by dependent products.

`dependson=[<PROJECT1>-<VERSION1>],[<PROJECT2>-<VERSION2>],...`
e.g. `abPackage-1.0.0,acPackage-1.0.4,PlotWS-1.0.1`

3. **Installation information:** The location of any files/data installed by the product. Considering this information will be used for the uninstall process, any change of the environment caused by the product has to be recorded in here. The directory needs to specify the parent directory if it is not an absolute path.
 - `location=[<DIRECTORY/FILE LOCATION>],[<DIRECTORY/FILE LOCATION>],...`
e.g. `/home/omii/OMII/jakarta-tomcat-5.0.25/webapps/abPackage,/home/omii/OMII/jakarta-tomcat-5.0.25/webapps/abPackageAdmin,/home/omii/OMII/CLINET/abPackage`
 - `webapp=nameOfAxisInstance` if the package has a separate webapp name rather than axis
 - `database=<nameOfDatabase1>,<nameOfDatabase2>,...` if the package creates databases (i.e. table spaces) within the OMII-UK DBMS

4.7 *Database Usage*

The OMII-UK server installation provides a DBMS, currently PostgreSQL, installed by the main OMII-UK stack installer. Access to this database can be obtained through the OMII properties (section 4.2) at install time and the service runtime environment. It is generalised such that any creation of tables and even database can be carried out during the server installation. Other access is normally carried out during service runtime.

We have mandated an Ant target called “**configure-db**” in the server binary build.xml for specifically creating tables and databases. Please see section 4.2 for the omii.mp.* properties that detail the locations and configuration of the OMII-UK DBMS. You can use these properties to connect to the DBMS.

4.7.1 *Table Namespace in an existing database*

The *mp* database is available to all of the commissioned software components. To avoid any namespace clashes in tables the following schema MUST be used:

<MP_PROJECT_NAME>_<TABLE_NAME>

For example, if the GridSAM wishes to use a table called 'jobs' the table name would be: GRIDSAM_jobs

Please also note that this database includes tables from other software components. They are meant for light-weight administration purposes. If your services require heavy weight database access you are advised to create your own database. Please see next section.

4.7.2 Database Namespace

Commissioned software packages that need to create their own database should also follow the following schema to avoid any namespace clashes:

<MP_PROJECT_NAME>_<DB_NAME>

For example, if the GridSAM wishes to use a database called 'admin' the database name would be: gridsam_admin.

To create a database, the recommended method is to use the omii.mp.* properties, including the username and password of the mp database user, to connect to the database first. Databases can be created from this location, because the mp user is permitted that right.

5 Documentation requirements

5.1 Introduction

It is difficult to provide strict rules for writing the documentation that accompanies a software component, because the structure and content of the documentation is ultimately dependent on the functionality of the software and the requirements of the intended user. However, there are some basic rules that can be applied to any documentation, as discussed below. Adhering to these basic rules will ensure the straightforward integration of the new documentation into the OMII's established documentation.

5.2 Location

The documents should be stored in a location that satisfies the above Distribution Structure. The documentation should not be duplicated in all three directories i.e. server-side documents should not be stored in the client and vice versa.

5.3 Format

Preferably, documentation should be provided in both HTML and PDF formats. For PDF, the page size should be A4.

5.4 High-level considerations

In general, the following questions should be addressed before writing the documentation:

1. What problem does the software solve?
2. How does the software solve the problem?
3. How can a user quickly access the functionality of the software?
4. How can a developer make use, or extend the functionality, of the software?
5. What is the current release state, and how does the current release differ from earlier releases?

The documentation should provide the answers to all of the above questions.

5.5 Structure

To ensure clarity, a strict linear structure should be employed throughout the documentation. The simplest way to ensure a linear structure is to separate the documentation into:

1. An introduction, including:
 - a description of the particular problem, software component, issue etc.
 - a summary of the subject matter that will be addressed in the following body section
 - sufficient detail to allow a user to quickly determine the location of information that is relevant to their query or problem
2. A body section that discusses the problem, software component, issue etc. in detail
 - the body section may be separated into chapters or sub-sections, the number of which will be dependent upon the complexity of the subject being discussed
 - the structure of the body section should follow that listed in the introduction
3. A summary of the salient points raised in the body section
 - a conclusion is not always appropriate, but one should be contemplated if particularly complex issues have been addressed

The linear structure should be applied both to the documentation as a whole, and to each chapter/section/sub-section within the documentation.

The number, order and naming of the sections comprised in the documentation is largely dependent upon the type of software that is being described e.g. the “introduction”, “installation guide”, “user guide”, and “tutorials” structure lends itself to end-user-centric software, the “requirements”, “function specification”, “design statement”, “java doc”, “junit report” structure is useful for developer-centric middleware, and the “readme”, “configure”, “autorun”, “help” structure is useful for GUI based applications. Thus, it is difficult to provide a universal structure for use in all documentation. However, we recommend reading Chapter 5 of “NASA Software Engineering Requirements” (issued by NASA), which provides a thorough overview of the sections that should be contemplated:

http://nodis3.gsfc.nasa.gov/npg_img/N_PR_7150_0002_/N_PR_7150_0002_.pdf

5.6 Content

The content of the documentation will vary, dependent on the type of software and its purpose. However, certain subjects are universal. The OMII require that the following subjects be included in all documentation:

- Summary
 - project name, language, software domain (service, client, tools)
 - what the software does and why the user would want to use it
 - project and technical contact information
- Getting Started
 - a short ‘read me’ or a getting started/quickstart guide with a link to the details of software pre-requisites, required elements of the OMII distribution, user guides, etc.
- Product Overview and Architecture
 - requirements specification, e.g. system overview, user cases, functional specification, relationships to external components, security requirements, design and implementation constraints, specifications, standards
 - software design description, e.g. design decisions and rationale, architecture design, decomposition and interrelationship between components, how the design satisfies the requirements, components input and output, overview of data flow and control flow

-
- interface design description, e.g. types of interface (data transfer or control), format of individual data elements provided by interfaces, protocols used by the interface, interface compatibility
 - Installation & Use
 - detailed installation and configuration guide
 - user guides
 - administration guides
 - details for obtaining and configuring any required third-party components
 - Support
 - tutorials and examples
 - sample applications & development guides
 - API documentation
 - 1-3 promotional slides in the Powerpoint format that provide a brief overview of the product
 - These slides should be suitable for disseminating to others.
 - Testing and Quality Assurance
 - documentation of testing strategy
 - description of test environment (operating system etc.)
 - instructions detailing how to run the tests
 - test results and analysis
 - overall assessment of the software, as demonstrated by the test results
 - any remaining deficiencies, limitations or constraints detected by testing

5.7 Additional Comments

References to external sources, hyperlinks etc. should be included, where possible, to allow the user to obtain further information if required.

Cross-referencing to relevant sections of the documentation should be used wherever possible. It is particularly important to implement cross-referencing when listing long sets of instructions, or when discussing complex inter-related concepts that are described in different sections of the documentation.

5.8 Style Guide

The following basic style guide is provided to ensure that the style of the documentation is compatible with that used in the OMII's established documentation.

5.8.1 Text

The text used in the documentation should comprise:

1. a maximum of three header sizes, all in the same font (e.g. Arial, left aligned, one line space after each paragraph)
2. one font for body text, which should be the same as the header font (e.g. Arial, Justified, one line space after each paragraph)
3. one constant-width font for code (e.g. Courier, left aligned, no line space after each paragraph)

5.8.2 Conventions

Where possible, the conventions used in OMII's documentation should be repeated in the documentation i.e.

[UNIX] denotes Unix specific commands
[Windows] denotes Windows specific commands
Bold used for file and directory names

5.8.3 Graphics

Where possible graphics should be supplied in the png, jpg or gif format.

6 Release check list

Before a software package is contributed to OMII-UK, we encourage software providers to review this following three step checklist.

6.1 Code check - Building the binary distributions from source

This should be a fully self-contained process where possible. If the build requires external third party components that cannot be supplied within the source distribution (possibly for licensing reasons), then clear (and preferably short!) instructions should be given that detail where to get those components and how to integrate them into the build process. If possible, these components should be simply 'plugged-in' in their distributable form in the build directory.

6.1.1 Verification Using the Automated Build Software

There exists a 'Commissioned Software Process and Test Service' distribution that includes the actual build infrastructure used within OMII-UK, to build/test source distributions and build client and server binaries. Full usage documentation is available in the release, which can be obtained from the OMII-UK repository for Commissioned Software partners. However, a more manual approach for verifying a source distribution is detailed below. The release also includes an exemplar source distribution called MPTestService, which utilises the MP database support detailed in this document.

6.1.2 Manual Build Verification

To verify, unpack the project's source, then with Apache Ant 1.6.1 installed, invoke the build in the following manner in the unpacked source directory:

```
$ ant build-all
```

The build process should produce, in the directory *<unpacked_source_directory>/dist*, two subdirectories: client and server, which each contain the appropriate binary distribution (see directory structure in section 2.3).

The OMII-UK build process will automatically produce two archives following the build. The two archives that are generated are:

```
<project-name>-<3-digit-version>-bin-server.tgz  
<project-name>-<3-digit-version>-bin-client.zip
```

For binary install verification (see the next two sections) these archives will need to be generated manually.

6.1.3 Verifying the Server Binary via Installation

The .tgz server distribution that is generated by the build process should be directly pluggable into the OMII-UK distribution MP server install process, by simply copying the server's .tgz into the *omii-server/managed_programme* directory of an unzipped OMII-UK server distribution (of an appropriate version).

After copying the server distribution, to verify, invoke the OMII-UK server stack installer and select 'Install Software Components'. Ensure that at least the following table entries are satisfied – i.e. they can be ticked as having fulfilled that test:

Environment	OMII-UK Stack Installation	
	Base/Extensions	Base/Extensions/Services
Root		
Non-root		

In addition, verification on the latest supported 1.4.2 JDK and 1.5.0 JDK in particular is strongly recommended.

6.1.4 Verifying the Client Binary via Installation

The .zip client distribution generated by the build process should be directly pluggable into the OMII-UK distribution MP client install process, by simply copying the client's .zip into the *omii-client/managed_programme* directory of an unzipped OMII-UK client distribution (of an appropriate version).

After copying the client distribution, to verify, invoke the OMII-UK client Managed Programme installer.

Verification on the latest supported 1.4.2 JDK and 1.5.0 JDK in particular is strongly recommended.

6.2 Document check

Check that all the documents are considered, that they are in the source release and that they are produced correctly in the generated binaries.

6.3 Packaging check

Package name, version

Document structure

Top level files