Project number: RI 211727

Project acronym: EDGeS

Project full title: Enabling Desktop Grids for e-Science

Research Infrastructures
INFRA-2007-1.2.2 Deployment of eInfrastructures for scientific communities

EDGeS Application Packaging for XtremWeb-HEP

Author: Oleg LODYGENSKY, Gabriel CAILLAT

Version: Initial draft

The EDGeS project has received Community research funding

1 Table of Contents

2 Status and Change History

Status:	Name:	Date:	Signature:	
initial	V2	4/11/2009	Gabriel CAILLAT	

Version	Date	Pages	Authors	Modification	

3 Glossary

XWHEP	(XtremWeb by High Energy Physics) Open source middleware for building volunteer computing and desktop grid projects. XWHEP is developed by IN2P3 and is based on XtremWeb developed by INRIA.
DG	(Desktop Grid) Variant of the volunteer computing model in which an organisation that manages several desktop PCs (an enterprise, educational centre, etc.) uses them for setting up a distributed project. The different PC computes together for this project. In this model the organisation develops and ports its own applications and all DG clients should be managed automatically.

4 Introduction

4.1 About this document

The aim of this document is to describe the EDGeS application packaging process.

4.2 Overview

EDGeS application bundles are used by two key persons in the application development lifecycle: the application developer and the application deployer.

The application developer is the person who creates application modules and assembles them into an EDGeS application bundle file.

The application deployer and administrator is the person who configures and deploys the EDGeS application to an XWHEP server, administers the computing and networking infrastructure where EDGeS applications run, and oversees the runtime environment. During configuration, the deployer follows instructions supplied by the application developer.

The document describes application packages both from the application developer and the application deployer point of view.

The first part of the document is devoted to application developers and describes the expected format of the XWHEP bundle belonging to a validated application stored in the EDGeS Application Repository (AR). We do not focus on the way applications are organized within the EDGeS AR.

The second part of the document is for XWHEP site administrators who would like to deploy an application to their DG server. It describes the process of deploying an EDGeS application bundle to a XWHEP server.

5 Creating application packages

Each application package must be created following the instructions in this document. The INSTALL file (see Section 5.9) contains the document version (e.g. "`Based on EDGeS Application Packaging Document v0.2`") which was used to create the bundle, see Section 2 for the current document version.

Every application has a **short name**, e.g. dsp. From this point on we will refer to this name with *shortname*.

Every XWHEP application must have **binaries** for each platform it can be executed. Every XWHEP application may have **libraries**, default **environment**, base **environment** and/or default **input**.

Every XWHEP application may come with a **master** like part. This part is responsible for generating workunits on the XWHEP, and for processing the results. This part may consist of different components: workunit generator(s) and validators. The common name of these components is **daemons**. The master part is typically a shell script calling the XWHEP client and is provided by the application developer.

5.1 Package file format

As the EDGeS Application Repository can store at most one file belonging to an application within a given host, all the files belonging to the validated application have to be stored in one file. The format of this file must be a **tar.gz** archive as such an archive can be easily handled on basically every operating system with an adequate compressed archive handler application. The name of this archive file must be ***shortname_version*.tar.gz**, e.g. *myapp_1.00.tar.gz*. The versioning must follow these rules:

- must have the format *X.YY*, where X is the major version consists of at least one digit, YY is the minor version consist always of two digits. A version number should look like e.g. *6.01* or *5.10*
- whenever the content of bundle is changed, the version number must be increased
- minor version number has always two digits, so e.g. after *0.99* comes *1.00*

The XWHEP bundle must have the following directory structure and must contain the following files:

`/binaries` . The `binaries` directory contains binaries of the application for all platforms it supports.

`/libraries` . The `libraries` directory contains optional libraries of the application for all platforms it supports.

`/baseenv` . The `baseenv` directory contains optional base environment of the application for all platforms it supports.
If set, the base environment is sent with all jobs.

`/defaultenv` the `defaultenv` directory contains optional default environment of the application for all platforms it supports.
If set, the base environment is sent with all jobs without specified environment.

`/defaultstdin` the `defaultstdin` directory contains optional default standard input of the application for all platforms it supports.
If set, the base environment is sent with all jobs without specified standard input.

`/master` the `master` directory contains optional master application of the application for all platforms it supports.
If set, the master application is to be paired with the XWHEP client.

`/README` the read me file contains general information about the application.

`/USECASE` this text file contains a use case of the application.

`/INSTALL` this text file contains information about the installation of the application package.

`/CHANGELOG` this text file contains information about history of the package.

Now we will describe each component in detail.

5.2 Binaries directory (mandatory)

As written above, the `binaries` directory contains every file and description belonging to the binaries of the application. This directory has to contain the following subdirectories and files:

platform$_1$ directory of the first supported platform, e.g. `i686-pc-linux-gnu`

`...` directories for other platforms

platform$_n$ directory of the last supported platform, e.g. `windows_intelx86`

5.3 Libraries directory (optional)

As written above, the `libraries` directory contains, if any, every file and description belonging to the libraries of the application. This directory has to contain the following subdirectories and files:

platform$_1$ directory of the first supported platform, e.g. `i686-pc-linux-gnu`

`...` directories for other platforms

platform$_n$ directory of the last supported platform, e.g. `windows_intelx86`

5.4 Base environment directory (optional)

As written above, the `baseenv` directory contains, if any, every file and description belonging to the base environment of the application. This directory has to contain the following subdirectories and files:

platform$_1$ directory of the first supported platform, e.g. `i686-pc-linux-gnu`

`...` directories for other platforms

platform$_n$ directory of the last supported platform, e.g. `windows_intelx86`

5.5 Default environment directory (optional)

As written above, the `defaultenv` directory contains, if any, every file and description belonging to the default environment of the application. This directory has to contain the following subdirectories and files:

***platform*$_1$** directory of the first supported platform, e.g. `i686-pc-linux-gnu`

... directories for other platforms

platform$_n$ directory of the last supported platform, e.g. `windows_intelx86`

5.6 Default standard input directory (optional)

As written above, the `defaultstdin` directory contains, if any, every file and description belonging to the default standard input of the application. This directory has to contain the following subdirectories and files:

***platform*$_1$** directory of the first supported platform, e.g. `i686-pc-linux-gnu`

... directories for other platforms

platform$_n$ directory of the last supported platform, e.g. `windows_intelx86`

5.7 Master directory

Optional
As already explained, the `master` directory contains every master-side component of the application. The master has to be paired with the XHWEP Client
This directory has to contain the following subdirectories:

***platform*$_1$** first supported platform as defined in Section **??** (e.g. `i686-pc-linux-gnu`)

... other platform directory

platform$_n$ last platform directory

And may contain the following optional directory:

source optional directory for source of the deamon

Note: master-side components are executed on your server, thus they need to be deployed only for the platform that matches the platform of your server, e.g. i686-pc-linux-gnu if you are running 32bit Linnux. Each *`platform`* directory contains the following subdirectories and files:

***daemon*$_1$** first deamon's directory

... other daemon's directory

daemon$_n$ last daemon's directory

dcapi-*shortname*.conf DC-API configuration file for master applications using DC-API. The requested format of this file is available through the documentation section of www.desktopgrid.hu at `http://www.desktopgrid.hu/storage/dcdoc/backends.html\#boinc`

***shortname*_master.xml** master-side description

The optional *`source`* directory must contain the following directories:

daemon$_1$ directory contains first deamon's source code

`...` directory contains other deamon's source code

daemon$_n$ directory contains last deamon's source code

5.7.1 Contents of a daemon directory

Within the *platform* directory The contents of a used daemon's directory isn't fixed. The recommended way is to place every binary and configuration file used by the given daemon within this directory. However bundle creators must take care so that the relevant file sections of the *`shortname`*`_master.xml` file reflect the structure of this daemon directory.

Within the *source* directory The contents of a used daemon's directory isn't fixed. The recommended way is to use Autotools[1] to configure the source code for building.

5.8 The README file

The **README** file contains general information about the application package and the application contained in the package. The contents are not defined but the file should typically include one or more of the following:

- general description of the application
- copyright and licensing information
- contact information for the distributor or programmer
- known bugs
- troubleshooting
- credits and acknowledgments

[1] See http://en.wikipedia.org/wiki/GNU_build_system

5.9 The INSTALL file

The **INSTALL** file should contain a step-by-step installation guide for the application deployers. Application deployment is done by the administrator of the XWHEP platform, using the XWHEP client.
A default **INSTALL** file can be found in the following section. If the deployment of the application requires any other steps to be done by the application deployer, or the installation process is different than the one described in the example INSTALL file, the contents of the example file should be replaced with detailed instructions reflecting the actual deployment process.

5.9.1 Example INSTALL file

```
Installation instructions for <YOUR APPLICATION>
Based on EDGeS Application Packaging Document v0.2

Copyright (C) 2009 <YOUR ORGANISATION>

<LICENSING INFORMATION>
This program is free software;
you can redistribute it and/or modify
it under the terms of the GNU General Public License version 2
as published by the Free Software Foundation.

*************************************************************************

SUPPORTED PLATFORMS:
  <UPDATE THE LIST OF SUPPORTED PLATFORMS>
  (1) Linux 2.2+
  (2) Mac OS X
  (3) Windows (Win 2K and higher)

SUPPORTED PROCESSOR ARCHITECTURES:
   Architectures known to work include Intel x86, Alpha, Sparc, Amd64, and ARM.
   <UPDATE THE LIST OF SUPPORTED ARCHITECTURES>

REQUIRES:
  <LIST ANY DEPENDENCIES>

OPTIONAL (but recommended):
  <LIST ANY OPTIONAL PACKAGES>

*************************************************************************

INSTALLATION COMMANDS FROM PACKAGE:

1.) extract the application package into a temporary directory

2.) to deploy the application's binaries run the following command in the appropriate binaries

    xwsendapp appName cpuType osName URI | UID : this sends/updates application; URI or UID po
```

```
************************************************************************

PLATFORM SPECIFIC NOTES:

<IF ANY OF THE SUPPORTED PLATFORMS REQUIRES SPECIAL HANDLING, UPDATE
THE FOLLOWING LIST>
* Linux:
* Mac OS X:
* Windows:
```

5.10 The CHANGELOG file

CHANGELOG must consist of one (for each bundle version) of the following entries:

```
version <VERSION> -- <DATE>

 * <TEXT>
 * <TEXT>

    -- <AUTHOR_NAME> <<AUTHOR_EMAIL>>
```

And must follow these criteria:

- Each entry should be separated by two blank lines from each other.
- <VERSION> is the bundle version described in Section 5.1
- <DATE> must be in RFC-2822 format, e.g. *Sat, 17 Oct 2009 11:32:31 +0200* (on Linux you can use *date -R* command to print the current date using this format)
- <TEXT> summarizes a change since the last version. Use separate ** <TEXT>* lines to describe each change in the bundle.
- Since each <TEXT> is a sentence, every one should end with a dot ('.').
- When updating CHANGELOG, the entry for latest version must be put at the top of the file.

5.10.1 Example CHANGELOG file

```
Version 0.02 -- Mon, 12 Oct 2009 16:13:43 +0200

 * New windows 32bit binary (3.02), compiled using mingw32.

    -- MAROSI Attila Csaba <atisu@sztaki.hu>


Version 0.01 -- Sat, 10 Oct 2009 11:37:20 +0200

 * Fixed dc-api file creation: mistyped name.
 * Added linux i686 and win32 binaries.
 -- MAROSI Attila Csaba <atisu@sztaki.hu>
```

6 Deploying application packages

EDGeS application packages have a standardized format to ease the deployment process. Application deployers should follow the instructions in the **INSTALL** file.
Application deployment is done by the administrator of the XWHEP platform, using the XWHEP client.
If the application package conforms to the EDGeS application bundle format, the installation steps are the following:

1. Download the application bundle from the EDGeS Application Repository
2. Extract the application package into a temporary directory
3. to deploy the application's binaries run the following command in the appropriate binaries/<PLATFORM>/ directory:
   xwsendapp appName cpuType osName URI | UID : this sends/updates application; URI or UID points to binary.

Always read the INSTALL file of the package before starting the installation! If the application deployment does not follow the standard installation procedure, that file should contain the required instructions.