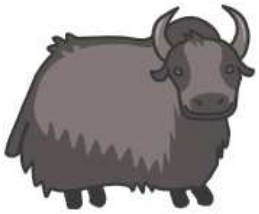
YAIM 4 reference guide



YAIM

Introduction

Yaim (Yaim Ain't an Installation Manager) is software to configure Grid services, but YAIM can also be used as a **general purpose** configuration tool completely independent from the Grid middleware. The aim of YAIM is to provide **simple** configuration methods that can be used to set up uniform Grid sites but can also be easily adapted to meet the needs of larger sites. To ensure that all system administrators can adapt YAIM to the local requirements, it has been implemented as a set of **bash scripts**. To support the EGEE project's component based release model **YAIM 4** has been **modularized** and a YAIM core is supplemented by component specific scripts, distributed as separate rpms.

YAIM's directory structure

When a **YAIM 4** module is installed the following directory structure is created under `/opt/glite/yaim`:

- `/functions/`: Contains the functions to configure the node types. They are all bash scripts.
- `/functions/local/`: Site administrators can put here their own function definition files. These will overwrite the default ones coming with the YAIM rpms. The file name has to be the same as the function name defined inside it.
- `/functions/pre/`: Function definition files having the same name as the original YAIM function but the defined function inside them has the '`_pre`' suffix. If this function exists, will be executed before the main function.
- `/functions/post/`: Function definition files having the same name as the original YAIM function but the defined function has the '`_post`' suffix. This function - if it exists - will be executed after the main function.
- `/node-info.d/`: Containsa file for each node type (configuration target) which defines the set of functions to be executed during the configuration of that node type. The file names in this directory should be the lower-case variant of the node type (configuration target.)
- `/defaults/`: The filenames in this directory have the same format as in

/node-info.d/ with a .pre and/or .post suffix. They are sourced before and after the main site-info.def and their purpose is to give meaningful default values - if possible - to some of the variables used by the given module.

- /bin/: Contains the main yaim executable.
- /log/: The location of YAIM's logfile, yaimlog.
- /examples/: This directory contains an example configuration storage. (i.e the main site-info.def file and the additional directories.) Its structure is explained in the next section.

YAIM's configuration storage

The /examples/ directory in /opt/glite/yaim is just a guideline and serious site admins should not store their configuration files in that location. YAIM allows having the site's configuration in a well separated and protected place. The **configuration** files of a site are to be stored in a **directory structure** having a fixed layout relative to the site's main site-info.def file. The site-info.def file has to sit in a directory (ex.: /root/siteinfodir/) which could contain the followings:

- site-info.def: This is the main configuration file of the site, to be sourced first, other optional configuration files will overwrite the values defined here.
- services/: Site administrators can place files into this directory with their names having the same format as in /opt/glite/yaim/defaults. These files are sourced only when that given service is configured, thus enabling service-specific configuration settings.
- nodes/: Each file in this directory should be the fully qualified name of a host. Only when that host is to be configured, the corresponding file will be sourced which allows host-specific configuration steps to be performed.
- vo.d/: The entries in this directory are to be named as the lower-case version of a VO to be supported. The file can contain VO specific variables and will overwrite the ones defined in the main site-info.def file.
- users.conf, groups.conf: Usually they reside on the same level in the directory hierarchy where the main site-info.def does, but their location is configurable via the USERS_CONF and GROUPS_CONF variables.

For an example configuration structure see:

http://grid-deployment.web.cern.ch/grid-deployment/yaim/example_configuration.tgz

How to build your own YAIM module

Adding your own YAIM module, configuration target (ex.: `LEMON_server`) is really simple! You only have to

- create your function list definition (`node-info.d/lemon-server`) :

```
LEMON_server_FUNCTIONS='config_lemon1,
config_lemon2,
config_lemon3'
```

- your functions:

```
functions/config_lemon1
functions/config_lemon2
functions/config_lemon3
```

- and an example service specific configuration file which contains the set of parameters used by your module, `examples/siteinfo/services/glite-lemon-server`, ex.:

```
LEMON_HOST=lemon.mydomain.yy
LEMON_LOG_LEVEL=8
LEMON_SEND_EMAIL=yes
LEMON_SUPPORT_EMAIL=lemon-support@mysite.yy
#LEMON_WS_HOST=lemon2.mydomain.yy
```

- Optionally you can specify a `lemon-server.pre` and `lemon-server.post` file in the `defaults/` directory, where you can set some default value. In `.pre` you can set the ones which have independent default values and ensure that the already defined environment variables are not affecting your configuration:

`defaults/lemon-server.pre:`

```
LEMON_LOG_LEVEL=6
LEMON_BIN_PATH=/usr/local/lemon/bin/
unset LEMON_SUPPORT_EMAIL
```

while in `.post` the ones which are not compulsory but whose value depends on some other 'lemon' variable:

`defaults/lemon-server.post:`

```
LEMON_WS_HOST=${LEMON_WS_HOST:-$LEMON_HOST}
```

- Comment your code ! Each line which has the following form:

```
####@ This is my comment to be displayed
```

will be displayed by the yaim -e command. This is very useful for site admins who would like to know what the functions are doing, but don't want to browse the code.

- For each of your functions `config_lemonx`, you can define two (optional) additional functions (in the same file as the main one), which have the same name as the main function but with the `'_check'` and the `'_setenv'` suffix. The purpose of the `'_check'` function is to check whether all the necessary variable used by the main function is defined:

```
config_lemon1_check:
function config_lemon1_check() {
   requires $1 LEMON_HOST LEMON_LOG_LEVEL
}
```

while in the `'_setenv'` function you should perform those configuration steps which affect the environment settings:

```
function config_lemon1_setenv() {
   yaimgridenv_set LEMON_HOST '${LEMON_HOST}'
   yaimgridpath_prepend PATH '${LEMON_BIN_PATH}'
}
```

These functions are automatically called by the main `yaim` executable when performing configuration (`yaim -c`) or when running a function (`yaim -r`) you don't have to include them in the function definiton list. (See the description of the yaim command.)

Create your rpm and don't forget to set a dependency on `glite-yaim-core`. If you want all this in the official YAIM cvs, no problem, just contact us ! For an example module have a look at:

http://grid-deployment.web.cern.ch/grid-deployment/yaim/example_module.tgz

The bin/yaim command

```
Usage: /opt/glite/yaim/bin/yaim <action>  <parameters>
Actions:
       -c | --configure   : Configure already installed services.
                            Compulsory parameters: -s, -n

       -r | --runfunction : Execute a configuration function.
                            Compulsory parameters: -s, -f
                            Optional parameters  : -n
```

```
       -v | --verify      : Goes through on all the functions and checks that
                            the necessary variables required for a given
                            configuration target are all defined
                            in site-info.def by executing only the '_check'
                            functions.
                            Compulsory parameters: -s -n

       -d | --debug       : Sets logging level, overwrites the
                            YAIM_LOGGING_LEVEL variable defined in
                            site-info.def. Its value can go from 1 to 7.

       -e | --explain     : Doesn't perform configuration but explains what the
                            functions are doing by printing out the comments
                            having the '####@ This is my comment.' form found
                            inside the functions.
                            Compulsory parameters: -s -n

       -h | --help        : Prints out this help.

       -a | --available   : Prints out the available configuration targets
                            provided by the installed yaim module rpms.
                            Compulsory parameter:  -s

Parameters:
       -s | --siteinfo:   : Location of the site-info.def file
       -n | --nodetype    : Name of the node type(s) to configure
       -f | --function    : Name of the functions(s) to execute

Examples:
       Configuration:
        ./yaim -c -s /root/siteinfo/site-info.def -n SE_dpm_mysql -d 6

       Running a function separately:
        ./yaim -r -s /root/siteinfo/site-info.def -n LFC_mysql -f config_gip

       Verify your site-info.def:
         ./yaim -v -s /root/siteinfo/site-info.def -n SE_dpm_mysql
```

The configuration flow

When launching the configuration the different configuration files will be sourced and will overwrite each other in the following order (supposing that we store the site configuration in `/root/siteinfo/` directory and configure service `myservice` on host `myhost`, supporting VO `myvo`):

1. `/opt/glite/yaim/defaults/site-info.pre  (Y)`

2. /opt/glite/yaim/defaults/myservice.pre (S)
3. /root/siteinfo/site-info.def (S)
4. /opt/glite/yaim/defaults/site-info.post (Y)
5. /opt/glite/yaim/defaults/myservice.post (Y)
6. /root/siteinfo/services/myservice (S)
7. /root/siteinfo/nodes/mynode (S)
8. /root/siteinfo/vo.d/myvo (S)
9. /opt/glite/yaim/node-info.d/myservice (Y)

The files with (Y) come with YAIM, while the files with (S) could be added by the site admin. Then for each function myfunc defined in /opt/glite/yaim/node-info.d/myservice yaim executes the followings if they are defined:

1. /opt/glite/yaim/functions/pre/myfunc
2. /opt/glite/yaim/functions/local/myfunc (if exists) or /opt/glite/yaim/functions/myfunc
3. /opt/glite/yaim/functions/post/myfunc

The log file

YAIM logs its output to /opt/glite/yaim/log/yaimlog. Each time the yaim command is executed an entry is added to this logfile containing the

1. command executed,
2. the site-info.def and other config files used,
3. the time and date information,
4. the version of the installed yaim rpms.

followed by the whole output of the configuration. This makes it easy to debug and archive the performed configuration steps.

Contact and other useful info

The main YAIM page:

https://twiki.cern.ch/twiki/bin/view/EGEE/YAIM

The new YAIM home page

http://www.yaim.info

is in preparation. Advanced bash scripting guide and other links here:

http://lcg.web.cern.ch/LCG/Sites/the-LCG-directory.htm

YAIM contact e-mail address:

yaim-contact@cern.ch

YAIM's license

```
############################################################
# Copyright (c) Members of the EGEE Collaboration. 2004.
# See http://www.eu-egee.org/partners/ for details on the
# copyright holders.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
# either express or implied.
# See the License for the specific language governing
# permissions and limitations under the License.
############################################################
```

YAIM's logo – the yak

The **yak** (*Bos grunniens*) is a long-haired humped domestic bovine found in Tibet and throughout the Himalayan region of south Central Asia, as well as in Mongolia. But this has nothing to do with our yak.

The Jargon File Defines « yak shaving » as « *Any seemingly pointless activity which actually necessary to solve problems which solves problems, which several levels of recursion later, solves the real problem you're working on.* » This is a bit like installing and configuring grid middleware :-) YAIM aims to take the pain out of grid system administration, while providing another form of amusement :-)

The yak is not a gnu.

Notes: