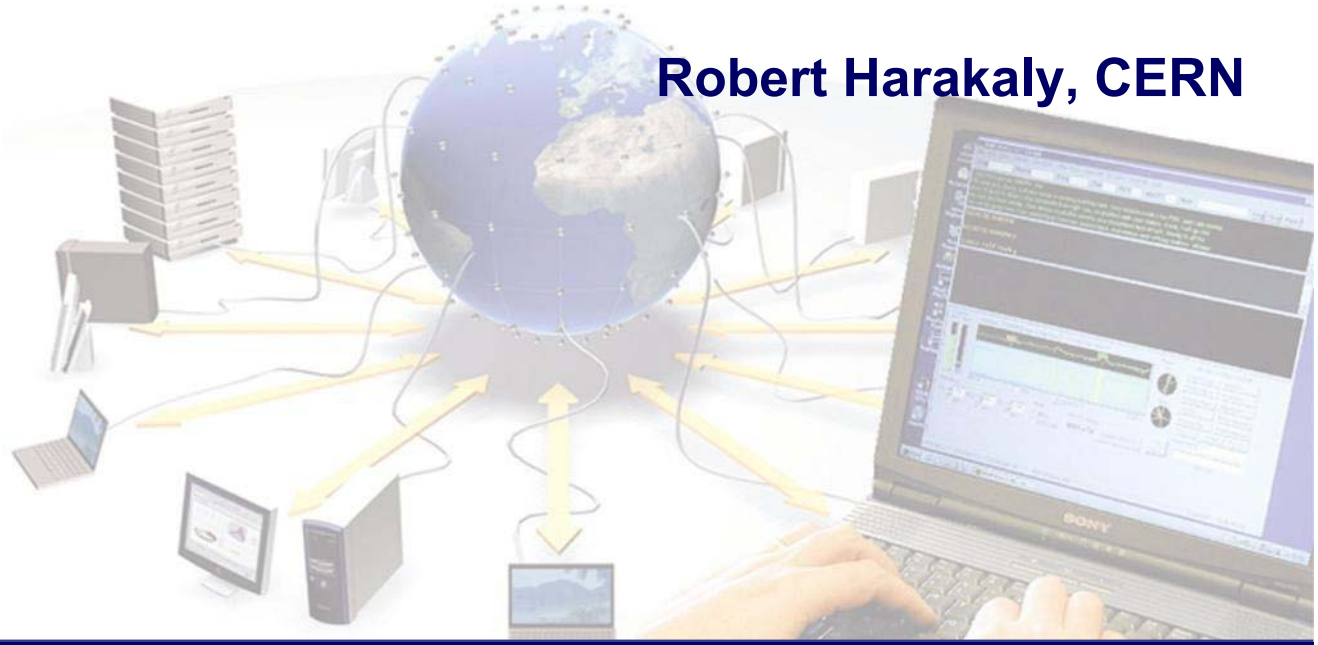
Enabling Grids for E-science in Europe

www.eu-egee.org

<Event>, <date>

Installation and configuration of gLite services

Robert Harakaly, CERN



EGEE is a project funded by the European Union under contract IST-2003-508833

Contents

- Configuration management
- Installation
- C/C++ services & clients

Problem

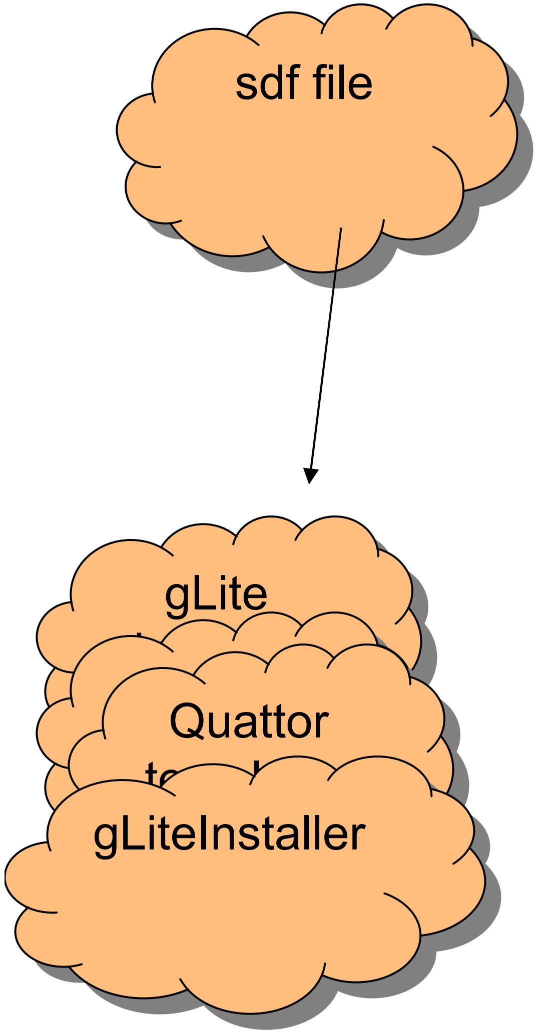
Enabling Grids for E-science in Europe

- We need to build structure supporting installation and configuration in:
  - Multi-platform environment (Linux, MS Windows)
  - Multi-package (RPM, tgz, MSI, …)
  - Supporting standard installation schemas (apt, yum, MSInstaller)
    - but not depending on any of them
    - Supporting additional installation as Quattor, etc.

Proposed solution

- Service based installation and configuration
  - Main building block is the service
  - Detailed description of the service and parameter control
    - Each service has its own XML description file (.sdf.xml) defining all needed information
    - Service description contains information (XML Schema) for configuration parameter checking
  - Any other description file (.spec, Quattor template, installation scripts, etc.) is derived from the service description file
  - Post-installation configuration done by using of the Python scripts
- It enables:
  - Controlled installation and post-install configuration using different methods starting from our gLiteInstaller through installation scripts and Quattor installation up to the manual installation

Service description file



Service description file

```
<service name="glite-io-service-client" version="N20041025"
  <components>
    <component name="glite-data-io-base"
      version="1.1.0" age="1" build="1" arch="i386"/>
    <component name="glite-data-io-quanta"
      version="1.0.0" age="1" build="1" arch="i386"/>
    <component name="glite-data-io-client"
      version="1.1.0" age="1" build="1" arch="i386"/>
    <component name="glite-data-config-service"
      version="1.1.0" age="1" build="1" arch="i386"/>
  </components>
  <dependencies>
    <external name="glite-essentials-cpp"
      version="1.0.1" age="1_EGEE" arch="i386"/>
    <external name="vdt_globus_essentials"
      version="VDT1.2.0rh9" age="1" arch="i386"/>
    <external name="gpt" version="VDT1.2.0rh9"age="1"
      arch="i386"/>
  </dependencies>
</service>
```

```
Summary: glite-io-service-client
Name: glite-io-client-service
Copyright: EGEE Copyright
Vendor: EGEE
Group:
Prefix: /opt/glite
BuildArch:
Requires: glite-data-io-base, glite-data-io-quanta, glite-data-io-client, glite-data-io-gss-auth,
glite-data-config-service, glite-essentials-cpp
Source:
```

```
template ... glite io client
#
# Copyright (c) Members of the EGEE Collaboration. 2004
# See http://eu-egee.org/partners/ for details on the copyright holders
# For license conditions see the license file or http://eu-egee.org/license.html
#
# glite-io-client installer template v. 0.2.0
#
# Glite dependencies
# glite io client dependencies
"/software/packages"=pkg_repl("glite-essentials-cpp","1.0.1-1_EGEE","i386");
"/software/packages"=pkg_repl("vdt_globus_essentials","VDT1.2.0rh9-1","i386");
"/software/packages"=pkg_repl("gpt","VDT1.2.0rh9-1","i386");
# glite-io-client RPMS
"/software/packages"=pkg_repl("glite-data-io-base","1.1.0-1","i386");
"/software/packages"=pkg_repl("glite-data-io-quanta","1.0.0-1","i386");
"/software/packages"=pkg_repl("glite-data-io-client","1.1.0-1","i386");
"/software/packages"=pkg_repl("glite-data-io-gss-auth","1.0.0-1","i386");
"/software/packages"=pkg_repl("glite-data-config-service","1.1.0-1","noarch");
```

Well Known installer script

```
#!/bin/sh
#                   gLite installation utility
# Copyright (c) Members of the EGEE Collaboration. 2004
# See http://eu-egee.org/partners/ for details on the copyright holders
# For license conditions see the license file or http://eu-egee.org/license.html
# glite-io-client installer v. 0.2.0
#
# The glite-io-client installer installs the gLite I/O Client
#
# Usage: glite-io-client-installer [-u|-v|--help]
#    -u      uninstall
#    -v      print version
#    --help  print script usage info
# Return codes: 0 - Ok
#    1 - if a file could not be downloaded
#####################################################################
# Parse the RPMLIST to strip out the RPMS that are already installed
function parseRPMList(){
   newRPMLIST=""
   localRPMLIST=`rpm -qa`
   for i in $RPMLIST
   do
     g=`echo $i | sed -e 's/\.i386\.rpm//g'`
     g=`echo $g | sed -e 's/\.noarch\.rpm//g'`
     if [ -z "`echo $localRPMLIST | grep $g`" ]; then
         newRPMLIST="${newRPMLIST} $i"
     else
         echo "$i is already installed. It will be skipped."
     fi
   done
```

Node Description file

```
<node name=”IO-FPS”>
  <description>
    gLite data IO and FPS node
  </description>
  <services>
    <service name="glite-io-server" version=”N20041025”>
      ...
    </service>
    <service name=”glite-fts”>
      ...
    </service>
    <service name=”glite-fps”>
      ...
    </service>
  </services>
  <depends>
    <!-- CA set -->
  </depends>
</node>
```

Service configuration file

```
<service name="glite-io-client-service" version=”1.1.1”>
  <parameters>
    <io-client.Server value=”io-server.cern.ch”/>
    <io-client.ServerPort value=”23456”/>
    <io-client.EncryptName value=”true”/>
    ...
  </parameters>
</service>
```

Service configuration file II

Enabling Grids for E-science in Europe

```
<service name=”VOMS-server”>
  <instance name=”HEP”>
    <parameters>
      <vo_name value=”HEP”/>
      <server_port value=”23456”/>
      <!-- parameters for HEP instance -->
      ...
    </parameters>
  </instance>
  <instance name=”BioMed”>
    <parameters>
      <vo_name value=”BioMed”/>
      <server_port value=”23457”/>
      <!-- parameters for BioMed instance -->
      ...
    </parameters>
  </instance>
</service>
```

XML Schema

```
<schema>
  <xs:annotation>
    <xs:documentation>
      gLite IO-client configuration parameters schema
    </xs:documentation>
  </xs:annotation>
  <xs:element name=”configuration”>
    <xs:complexType>
      <xs:sequence>
        <xs:element name=”io-client.Server”>
          <xs:annotation>
            <xs:documentation>
              Address of gLite IO server serving the client
            </xs:documentation>
          </xs:annotation>
          <attribute name=”value” type=”xs:string”/>
        <xs:element name=”io-client.ServerPort”>
          <xs:annotation>
            <xs:documentation>
              TCP port number of gLite IO server
            </xs:documentation>
          </xs:annotation>
          <attribute name=”value”>
            <xs:simpleType>
              <xs:restriction base=”xs:integer”>
                <xs:minInclusive value=”1”/>
                <xs:maxInclusive value=”65535”/>
              </xs:restriction>
            </xs:simpleType>
          </attribute>
        </xs:element>
```

XML Schema file is used for validation of the service configuration file, configuration parameters documentation

Post Installation Configuration

Enabling Grids for E-science in Europe

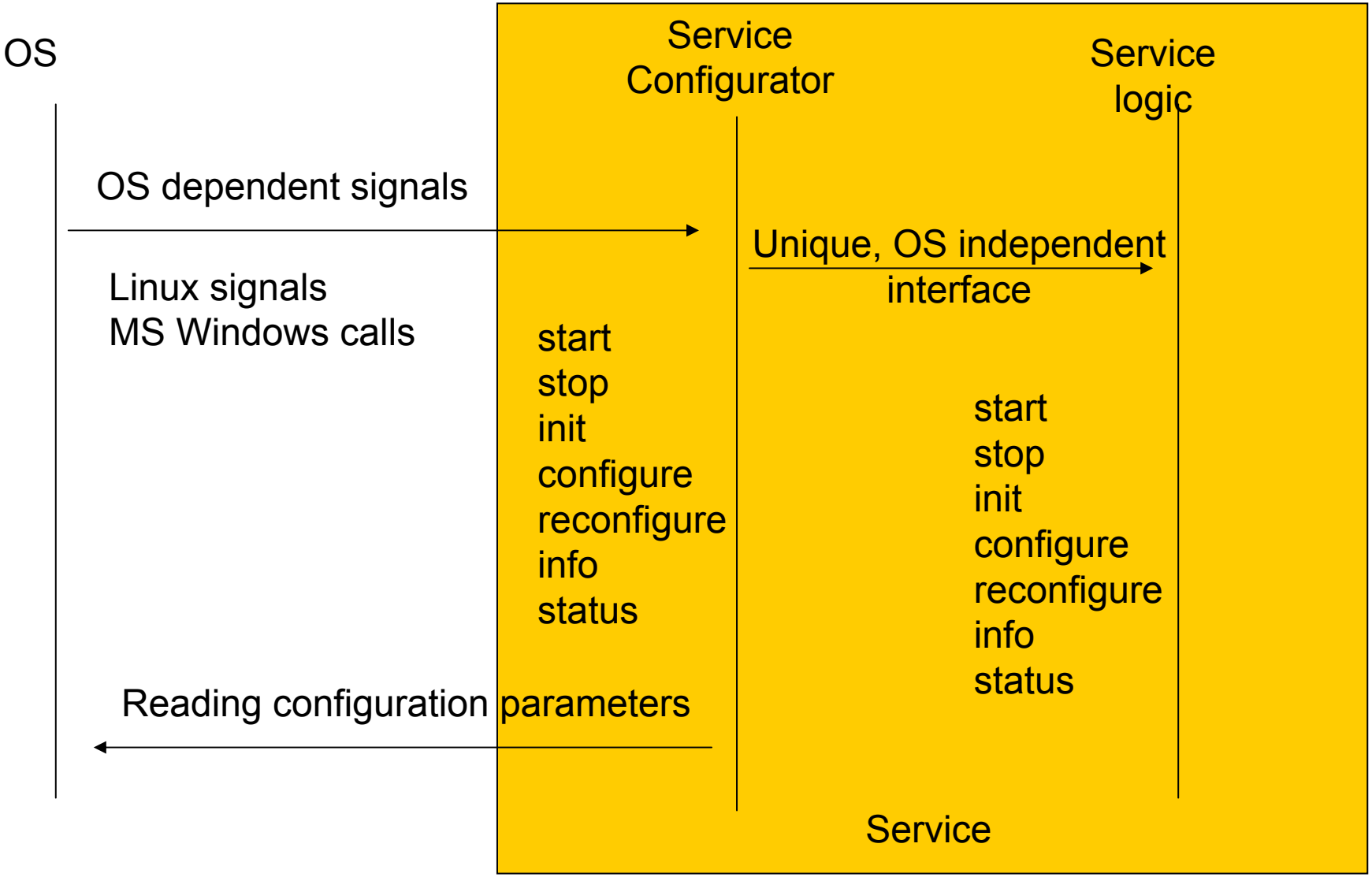
- Configuration parameters described in the service configuration file
- Configuration itself is done by Python modules
- All, distributed in the service deployment package (RPM)
- Configuration files location:
  - ${GLITE_LOCATION}/etc/config
    - glite-global.cfg.xml
    - glite-<service>.cfg.xml
  - ${GLITE_LOCATION}/etc/config/scripts
    - Python configuration modules for each service

C/C++ services and clients configuration

Enabling Grids for E-science in Europe

- The configuration of C/C++ services differs from the one of Tomcat services (see presentation of Joachim)
- Configuration done by using of “Service Configurator” as an internal part of the service
- Role of the Service Configurator :
  - Hide OS dependent issues from the service (messaging, …)
  - Create an unique interface for the management of the services
  - Load service configuration and forward it to the service
  - Enable addition of the monitoring functionality to the services, again by providing an unique interface to this

Interfaces


OS
Service Configurator
Service logic
OS dependent signals
Linux signals
MS Windows calls
Unique, OS independent interface
start
stop
init
configure
reconfigure
info
status
start
stop
init
configure
reconfigure
info
status
Reading configuration parameters
Service

Interfaces II

Enabling Grids for E-science in Europe

```
/** Initialize the Component. Parameters passed during the
  * Initialization are not supposed to change */
 virtual int init(const Params& params) = 0;
/** Configure the Component. This method is also use to reconfigure
  * the component after a stop. Parameters passed during the
  * Configuration could be modified on a reconfiguration */
 virtual int configure(const Params& params) = 0;
/** Reconfigure onfly (dynamic parameters) the Component.*/
 virtual int reconfigure(const Params& params) = 0;
/** Start the Component. This method is called to start the
  * execution or restart it after a reconfiguration */
 virtual int start() = 0;
/** Stop the Component. This method is called to stop the execution
  * either is case of shutdown and in case of reconfiguration */
 virtual int stop() = 0;
/** Finalize the Component */
 virtual int fini() = 0;
/** Pause Component. */
 virtual int pause() = 0;
/** Resume Component. */
 virtual int resume() = 0;
```

Parameters

Enabling Grids for E-science in Europe

- *Initialization:* all parameters which can be set up only during first initialization of the service and cannot be modified later. These parameters are mostly the parameters which controls the essential parts of the service as some memory allocation parameters, etc.
- *Configuration:* other parameters which can be modified during the lifetime of the service. To avoid the race conditions, configurator will stop the service before the modification of these parameters:
- stop – configure(with new configure parameters) – start.
- *Dynamic:* parameters that can be modified during service is running, in cases where the race conditions are not an issue. Typical example can be the debuglevel parameter

Client configuration

Enabling Grids for E-science in Europe

- C/C++ and JAVA clients can use simplified interface from the one on the slide 13.

Hot issues

Enabling Grids for E-science in Europe

- Post installation configuration
  - Actions
- Service configuration
  - Configuration parameters
  - Complex parameter types
- Service description
  - Service relations (mostly databases)