Enabling Grids for E-science in Europe

JRA1 Workshop Padova, 2004 November 15-17

Data Management Service Configurator

Paolo Badino

on behalf of the JRA1 DM Cluster:

Paolo Badino, Ricardo Brito da Rocha, Akos Frohner, Peter Kunszt, Gavin McCance, Krzysztof Nienartowicz, Jozsef Kovacs, Daniel Rodrigues

gLite

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

Contents

- ComponentConfigurator Pattern
- Data Management Service Configurator
  - Configuration files
  - Daemon Laucher
  - Config Generator
- Some examples
  - GLite I/O Server
  - GLite I/O Client
  - Unit Test Applications
- Improvements

ComponentConfigurator Pattern

Enabling Grids for E-science in Europe

- Component
  - Logical Module
  - Shared Library
- Service
  - Composition Components
  - Process

- Keys
  - Provide a common interface for configuration
  - Decouple configuration from application logic
  - Components are loaded at run-time
- Advantages
  - Uniformity
  - Simplified Administration
  - Modularity
  - Dynamic Configuration
  - Tuning

Configuration File

- XML
  - Flexible
  - Extensible

```
<service name="glite-io-server">
 <components>
  <component name="io-daemon">
   <lib>libglite_data_io_daemon.so</lib>
   <dependencies>
    <lib>libglobus_gssapi_gsi_gcc32dbg.so</lib>
   </dependencies>
   <init>
    <param name="Port"><value>9999</value></param>
   </init>
  </component>
  <component name="io-authz-fas">
   <lib>libglite_data_io_authz_fas.so</lib>
   <config>
    <param name="FasEndPoint">
     <value>http://lxb2024.cern.ch:8080/glite-data-catalog-service-fr/services/FiremanCatalog</value>
    </param>
   </config>
  </component>
  <component name="io-resolve-common">
   <lib>libglite_data_io_resolve_common.so</lib>
   <config>
    <param name="SrmEndPoint">
     <value>httpg://gridftp05.cern.ch:8443/srm/managerV1</value>
    </param>
    <param name="SeProtocol">
     <value>rfio</value>
    </param>
   </component>
   <component name="io-resolve-fireman">
    <lib>libglite_data_io_resolve_fireman.so</lib>
    <config>
     <param name="FiremanEndPoint">
      <value>http://lxb2024.cern.ch:8080/glite-data-catalog-service-fr/services/FiremanCatalog</value>
     </param>
    </config>
 </components>
</service>
```

Service Configurator

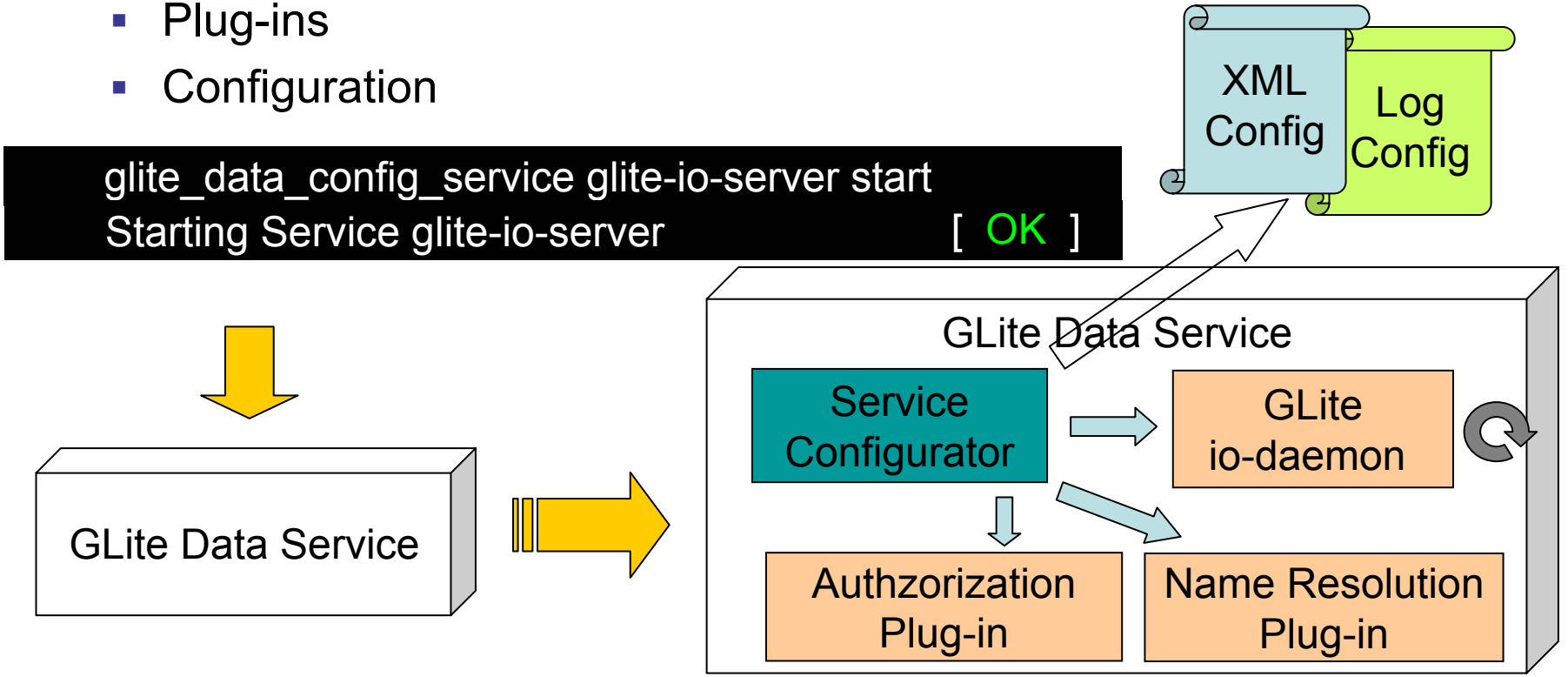
- ServiceConfigurator
  - Configure Log4Cpp (Using Log4J Property File)
  - Load all the components dynamically (Configuration File)
  - Configure all the Components
  - Notify the Components to start/stop
  - Handle Reconfiguration
- GLite Data Service a.k.a. Daemon laucher
  - Spawn a daemon process
  - Instantiate ServiceConfigurator and call it
  - Handle control signals
  - Monitor the daemon process
- GLite Data glite_data_config library
  - C library
  - Instantiate and manage ServiceConfigurator
  - Can be used to configure libraries or application that doesn't need to run as daemon or services

Independent from GLite I/O !

GLite I/O Server

- GLite I/O Server
  - Daemon
  - Plug-ins
  - Configuration

```
glite_data_config_service glite-io-server start
Starting Service glite-io-server                    [ OK ]
```



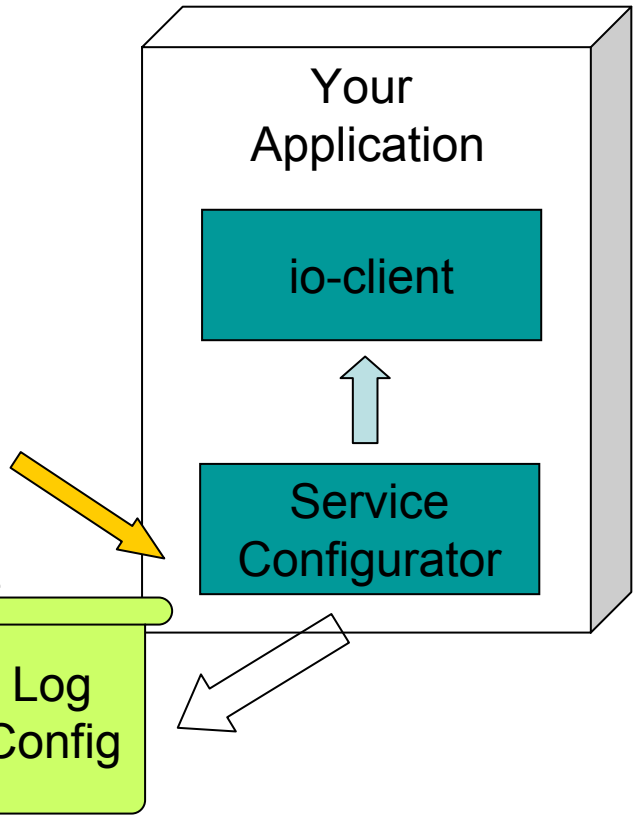
GLite I/O Client

- GLite I/O client
  - Library
  - Configuration

```
// Inside you client application:
glite_posix_open(filename,O_RDONLY,0);
```

```
// Inside glite_open:
glite_config_initialize("glite-io-client");
```

```
// Send a request to the GLite Io Server
```



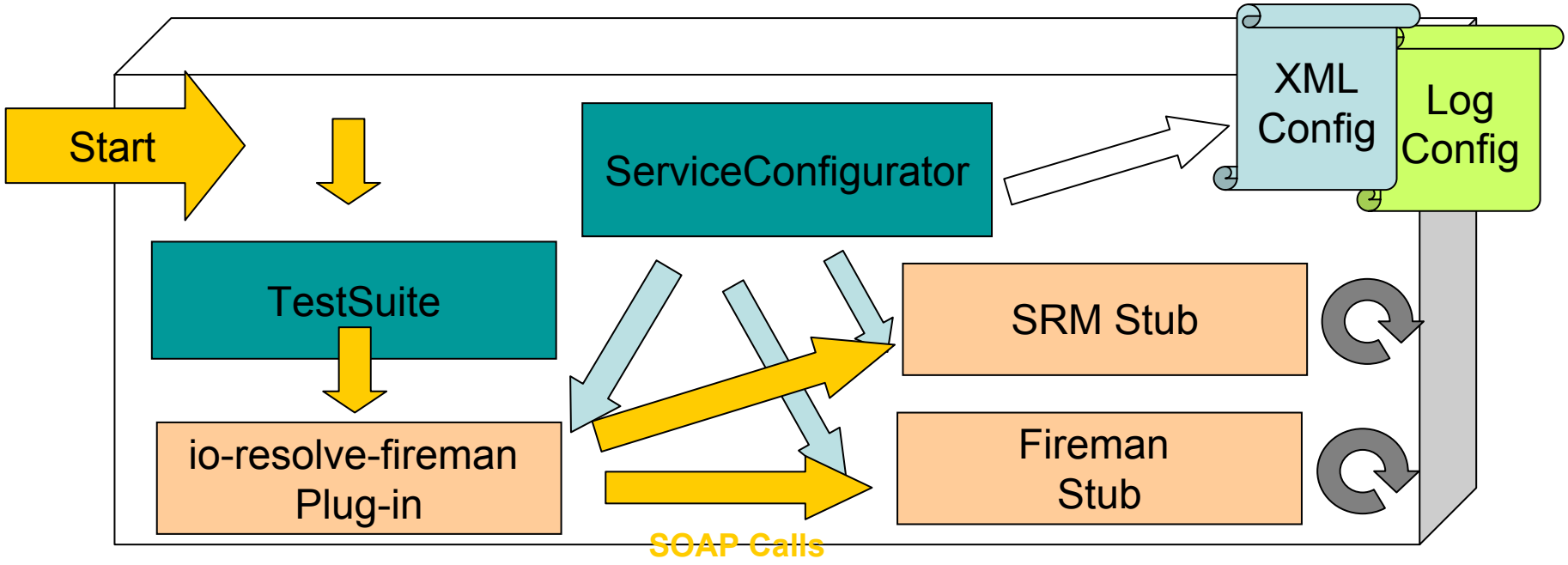
GLite I/O Client (2)

- Override GLite I/O Client default configuration

```
<service name="my-application">
 <components>
  <component name="io-client">
   <init>
    <param name="Server"><value>lxb1434.cern.ch</value></param>
    <param name="ServerPort"><value>9999</value></param>
    <param name="EncryptName"><value>true</value></param>
    <param name="EncryptData"><value>true</value></param>
   </init>
  </component>
  <component name="my-component">
   <lib>libmy_component.so</lib>
   <config>
    <param name="Param1"><value>A value</value></param>
   </config>
  </component>
 </components>
</service>
```

Unit Test Applications

- Example: org.glite.data.io-resolve-fireman Unit Test
  - GLite I/O server plug-in to interact with the Fireman Catalog and the SRM
  - Stub implementation (using GSOAP) of the Fireman and SRM interfaces dynamically loaded into the TestSuite application



Configuration Template

```
<service name="glite-io-server">
 <components>
  <component name="io-daemon">
   <config-template><description>The GLite IO Daemon represents the server side of the glite-io subsystem</description>
    <lib>libglite_data_io_daemon.so</lib>
    <dependencies>
     <lib>libglobus_gssapi_gsi_gcc32dbg.so</lib>
    </dependencies>
    <init>
     <param name="Port" mandatory="false" type="int" advanced="false"><description>The port to be used to contact the server</description>
       <value>9999</value>
     </param>
    </init>
   </config-template></component>
  <component name="io-authz-fas">
   <config-template><description>Implements the GLite-IO server Authz plugin</description>
    <lib>libglite_data_io_authz_fas.so</lib>
    <config>
     <param name="FasEndPoint" mandatory="true" type="string"><description>FAS catalog endpoint</description></param>
    </config>
    </config-template></component>
   <component name="io-resolve-common">
    <config-template><description>Provides some common functions for the resolve plugin</description>
     <lib>libglite_data_io_resolve_common.so</lib>
      <config>
       <param name="SrmEndPoint" mandatory="true" type="string"><description>The endpoint of the SRM Server</description></param>
       <param name="SeProtocol" mandatory="true" type="string"><description>The protocol to be used to contact the SE…</param>
     </config-template></component>
    <component name="io-authz-fas">
     <config-template><description>Implements the GLite-IO server Ressolve plugin</description>
      <lib>libglite_data_io_authz_fas.so</lib>
      <config>
       <param name="FiremanEndPoint" mandatory="true" type="string"><description>FiReMan catalog endpoint</description></param>
      </config>
     </config-template></component></components>
</service>
```

ConfigGenerator

- Parse The Configuration Template
-
```
glite_data_config_generator \
```
-
```
io-daemon.Port                    = 9999
io-authz-fas.FasEndPoint          = http://lxb2024.cern.ch:8080/glite-catalog/services/FiremanCatalog
io-resolve-common.SrmEndPoint = httpg://gridftp05.cern.ch:8443/srm/managerV1
io-resolve-common.SeHostName = gridftp05.cern.ch
io-resolve-common.SePort          = 8443
io-resolve-common.SeProtocol      = rfio
io-resolve-common.RootPath        = /castor/cern.ch/user/g/gproduct/EGEETEST/SE/
io-resolve-fireman.FiremanEndPoint = http://lxb2024.cern.ch:8080/glite-data-catalog/services/FiremanCatalog
log.Priority = DEBUG
log.fileName = /var/glite/log/glite-io-server.log
```
-
```
glite_data_config_generator \
 -f /opt/glite/share/config/glite-data-io-server/io-server.config.xml \
 -o /opt/glite/etc/glite-io-server.properties.xml \
 -l /opt/glite/etc/glite-io-server.log-properties \
 -p io-server.properties
```
- Generate the Configuration Files

Summary

- The configuration is stored in XML
  - Flexible, extensible configuration using the CC pattern
- Possible extensions
  - XML files contain ‘static’ values (you need to edit the file to reconfigure the service) => Imagine **dynamic** XML:
    - Parameters might be configured to be retrieved from the information services
    - Specify an RGMA value, BDII value, http, etc.
  - Support Complex Types: list, map, struct,..
  - User-extensible configuration
    - Allows user to override some system values
      - Read configuration from $(GLITE_LOCATION)/etc
      - Read user-specific values from $(GLITE_LOCATION_USER)/etc
  - And more…