Table of Contents

The CEMonitor configuration file for both EMI-2 and EMI-3 releases

Introduction

The CEMonitor service is the gLite component responsible for gathering and providing information coming from the Computing Element (CE). Although that is its main task, CEMonitor must considered as a general purpose notification framework able to collect and handle any kind of information even not GRID oriented. In fact its architecture based on the *sensor* concept, is very flexible and makes CEMonitor easily customizable by plugging new sensors for every specific information to be handled. For example it can be coupled with CREAM for publishing to the users the status changes of their own jobs, as well as in the context of the Open Science Grid (OSG) ReSS project, CEMonitor is used to collect resource information and send them to a central Information Gatherer in classad format. It is possible to interact with CEMonitor in a synchronous way (i.e. the client queries the service to get the required info) or asynchronously (i.e. the client can subscribe to get asynchronous notifications coming from CEMonitor). Moreover the administrator can configure predefined subscriptions, so that the specified clients are automatically notified with asynchronous events, without the need of an explicit subscription.

The CEMonitor configuration file structure

The default location of the CEMonitor configuration file is /etc/glite-ce-monitor/cemonitor-config.xml. It is a XML file composed of a set of elements basically of five different types:

- *sensor*: defines the capability for collecting and handling specific information
- *subscription*: used by the administrator so that the specified clients are automatically notified about a event with asynchronous notifications, without the need of an explicit subscription
- *action*: defines the specific action (e.g. SendNotification, DoNotSendNotification, SendExpiredNotification, etc) associated to a subscription to be executed whenever an event occur (e.g. new job status change)
- *queryprocessor*: defines the engine for executing queries (e.g. XPath, XQuery, SQL) against the published information
- *argus-pep* or *authzchain*: defines the security authorization layer based on ARGUS service or gJAF

```
<service id="main-service" {service attributes}>
    <sensor>
    ...
    <subscription>
    ...
    <action>
    ...
    <queryprocessor>

    <authzchain name = "chain-1">
          <plugin name = "the name" classname = "org.glite.ce.xxx">
              <parameter name = "name_1" value = "value_1" />
              ...
              <parameter name = "name_x" value = "value_x" />
          </plugin>
          ...
    </authzchain>
</sensor>
```

The sensor

Each topic published by CEMonitor is produced by a corresponding *sensor* which is a component responsible for actually generating events to be notified for a specific topic. Sensors can be plugged at runtime: whenever a new sensor is added, CEMonitor automatically generates the corresponding topic so that the users can subscribe to it. Each sensor can provide either asynchronous notification to registered listeners, or can be queried synchronously. The configuration of a sensor is based on a well defined structure composed of a set of mandatory attributes and properties (i.e. couples of name=value) which are specific for each sensor:

```
<sensor id="" name="" type="" jarpath="">
    <property name="name_1" value="value_1" />
    ....
    <property name="name_x" value="value_x" />
</sensor>
```

The following is the list of the mandatory attributes with their meaning:

- id: the string which identifies uniquely the sensor (*)
- name: the name of the sensor (e.g. "CREAM Job Sensor", "OSG CE Sensor")
- type: the name of the type to which the sensor belongs to (e.g. "CREAM_JOBS", "OSG_CE") (*)
- jarpath: the path of the jar implementing the sensor (*)

* please DO NOT change the default value which comes from the YAIM configuration.

The subscription

CEMonitor publishes information as *topics* and for each one, it maintains the list of *events* to be notified to the users. Topics can have three different levels of visibility: *public*, meaning that everybody can receive events associated with the topic; *group*, meaning that only member of a specific VO can receive notifi cations; and *user*, meaning that only the user which created the topic can receive notifications. Users can create *subscriptions* for topics of interest and each subscription is identified by a unique ID, has an expiration time and an update frequency *f*. CEMonitor checks every 1=f seconds whether there are new events for the topic associated to the subscription: if so, the events are sent to the subscribed users. Unless a subscription is explicitly renewed by its creator, it is removed after the expiration time and no more events will be notified. Special subscriptions may be created by the administrator so that the specified clients are automatically notified about a event with asynchronous notifications, without the need of an explicit subscription. Moreover specific policies and actions can be set on the subscription so that CEMonitor executes the proper action if the conditions satisfy the predefined policy or less. For example it is possible to define a subscription which notifies the client only when the CE queue is empty or the number of en-queued jobs is less than a well defined value).

The configuration of a subscription is based on a well defined structure composed of a set of mandatory attributes and elements which are specific for each subscription:

```
<subscription id="" monitorConsumerURL="" sslprotocol="" retryCount="">
    <topic name="">
        <dialect name="" />
    </topic>
    <policy rate="60">
        <query queryLanguage="ClassAd"><![CDATA[GlueCEStateWaitingJobs<2]]></query>
        <action name="SendNotification" doActionWhenQueryIs="true" />
        <action name="SendExpiredNotification" doActionWhenQueryIs="false" />
    </policy>
</subscription>
```

The following is the list of the mandatory attributes with their meaning:

- id: the string which identifies uniquely the sensor (*)
- monitorConsumerURL: the URL of the event consumer (i.e. the notification listener)
- sslprotocol: the optional attribute which defines the security level (e.g. "SSLv3")
- retryCount: represents the number of attempts to contact the consumer (if it is not reachable) before giving up. If -1 is specified, CEMonitor will never give up
- topic: the topic of interest (e.g. "CREAM_JOBS")
- topic dialect: a specific output format which can be used to render the events. This means that a sensor can publish information in different formats: for example, job status change information could be made available in Condor classad format or in XML format. When a user subscribes to a topic, she can also specify an appropriated dialect for rendering the notifications. The dialect is applied by CEMonitor before sending the notifications.
- policy: is a rule which instructs CEMonitor to executes proper actions
- policy rate: the policy is executed at fixed rate (default: *60* sec)
- policy query: the expression query expressed in a well defined queryLanguage (e.g. classad, xpath, sql)
- policy action: the action CEMonitor executes if the policy is satisfied or less (i.e. doActionWhenQueryIs= {true | false})

* please DO NOT change the default value which comes from the YAIM configuration.

The action

Whenever a new event is generated by the sensor, CEMonitor triggers an *action* on that event. In most cases, the action simply instructs CEMonitor to send a notifi cation to the user for that event. Of course, it is possible to extend CEMonitor with additional types of user-de ned actions:

```
<action id="" name="" type="action" jarpath=""/>
```

The following is the list of the mandatory attributes with their meaning:

- id: the string which identifies uniquely the sensor (*)
- name: the name of the action (e.g. "SendNotification", "DoNotSendNotificationr")
- type: the fixed value is "action" (*)
- jarpath: the path of the jar implementing the action (*)

* please DO NOT change the default value which comes from the YAIM configuration.

The query processor

This element represents the implementation of the query engine for the specific queryLanguage supported by the sensors (e.g. "xpth).

```
<queryprocessor id="" name="" type="queryprocessor" jarpath="" />
```

The following is the list of the mandatory attributes with their meaning:

- id: the string which identifies uniquely the sensor (*)
- name: the name of the queryprocessor (e.g. "ClassAd", "RegEx", etc)
- type: the fixed value is "queryprocessor" (*)
- jarpath: the path of the jar implementing the queryprocessor (*)

* please DO NOT change the default value which comes from the YAIM configuration.

The security authorization layer

CEMonitor supports two different authorization systems based on the ARGUS service or the gJAF (grid Java Authorization Framework). Therefore the CEMonitor configuration depends on the authZ system selected. In case of ARGUS the XML section looks like:

```
<argus-pep name="pep-client1"
    resource_id="CREAM_PEPC_RESOURCEID"
    cert="TOMCAT_HOSTCERT_LOCATION"
    key="TOMCAT_HOSTKEY_LOCATION"
    passwd=""
    mapping_class="org.glite.ce.monitor.authz.argus.ActionMapping">
    <endpoint url="ARGUS_PEPD_ENDPOINTS" />
</argus-pep>
```

while the configuration for the gJAF system is:

```
<authzchain name="chain-1">
    <plugin name="bannerpdp" classname="org.glite.ce.commonj.authz.gjaf.BlackListServicePDP">
        <parameter name="blackListFile" value="/etc/lcas/ban_users.db" />
    </plugin>

    <plugin name="admincheckpip" classname="org.glite.ce.commonj.authz.gjaf.AdminCheckerPIP">
        <parameter name="adminList" value="/etc/grid-security/admin-list" />
    </plugin>

    <plugin name="gridmappdp" classname="org.glite.ce.commonj.authz.gjaf.GridMapServicePDP">
        <parameter name="gridMapFile" value="/etc/grid-security/grid-mapfile" />
    </plugin>

    <plugin name="vomspdp" classname="org.glite.ce.commonj.authz.gjaf.VomsServicePDP">
        <parameter name="gridMapFile" value="/etc/grid-security/grid-mapfile" />
    </plugin>
</authzchain>
```

We suggest not to change the default values coming from the YAIM configuration phase. The parameters are all mandatory and their values well defined therefore there is no need to change them.

-- LisaZangrando - 2013-03-26

This topic: CREAM > CEMonitorConfigurationFile
Topic revision: r2 - 2013-03-26 - LisaZangrando