Component Configuration

There are a number of components that need be configured for the system to work. Such components are: implementations of the `ConfiguraitonEntry Factory` interface, Login Modules, Login Module Use among others.

We start with GBeans that implement `ConfigurationEntryFactory` interface and supporting GBeans, followed by the `LoginModule` gbean.

Configuring DirectConfigurationEntry

`DirectConfigurationEntry` exposes login module directly to JAAS clients. You have to specify Login Module here directly. To be able to login into Geronimo specify `JaasLoginCoordinator` login module.

`DirectConfigurationEntry` GBean declares following metadata:

- `applicationConfigName` - attribute; application configuration name; this is a key by which configuration entry is found.
- `controlFlag` - attribute; login module control flag according to the JAAS semantics; the only value that makes sense here is REQUIRED.
- `wrapPrincipals` - attribute; possible values are true and false. If set to **true**, all Principals generated by the login module (**Login Domain**) will be wrapped into the `DomainPrincipal` and every `DomainPrincipal` will be wrapped into the `RealmPrincipal`. This enables J2EE role mappings into `DomainPrincipals` and `RealmPrincipals`.
- `Module` - reference; This is object name specification for the `LoginModuleGBean`.

The following example shows how `DirectConfigurationEntry` is configured to use the `ServerLoginCoordinator` login module GBean.

```
<configuration
    xmlns="http://geronimo.apache.org/xml/ns/deployment"
    parentId="org/apache/geronimo/Client"
    configId="org/apache/geronimo/ClientSecurity"
    >
<GBean name="ServerLoginStubDCE" class="org.apache.geronimo.security.jaas.DirectConfigurationEntry">
        <attribute name="applicationConfigName">server-login</attribute>
        <attribute name="controlFlag">REQUIRED</attribute>
        <reference name="Module">                        <!-- reference to the login module GBean:
name=ServerLoginCoordinator -->
            <name>ServerLoginCoordinator</name>
        </reference>
</GBean>

<GBean name="ServerLoginCoordinator" class="org.apache.geronimo.security.jaas.LoginModuleGBean">
        <attribute name="loginModuleClass">org.apache.geronimo.security.jaas.client.JaasLoginCoordinator<
/attribute>
        <attribute name="serverSide">false</attribute>
        <attribute name="options">
            host=localhost                               <!-- Geronimo login service endpoint -->
            port=4242
            realm=geronimo-properties-realm               <!-- Security realm name -->
        </attribute>
        <attribute name="loginDomainName">geronimo-properties-realm</attribute>
</GBean>
</configuration>
```

Back to Top

Configuring ServerRealmConfigurationEntry

`ServerRealmConfigurationEntry` connects server side component (such as a Servlet) to the **Security Realm**. It allows **decoupling** of configuration name and **Security Realm** name.

`ServerRealmConfigurationEntry` declares following metadata:

- `applicationConfigName` - attribute; application configuration name; this is a key by which configuration entry is found.
- `realmName` - attribute; security realm name.
- `LoginService` - reference; object name for the JAAS Login Service GBean.
- `wrapPrincipals` - attribute; possible values are true and false. If set to **true**, all Principals generated by the login module (**Login Domain**) will be wrapped into the DomainPrincipal and every DomainPrincipal will be wrapped into the RealmPrincipal. This enables J2EE role mappings into DomainPrincipals and RealmPrincipals.

The following example shows how to setup the `ServerRealmConfigurationEntry` with the name of JMX. The security realm name is `geronimo-properties-realm`.

```
<configuration
    xmlns="http://geronimo.apache.org/xml/ns/deployment-1.0"
    configId="org/apache/geronimo/Security"
    parentId="org/apache/geronimo/RMINaming"
    >

    <GBean name="JMX" class="org.apache.geronimo.security.jaas.ServerRealmConfigurationEntry">
        <attribute name="applicationConfigName">JMX</attribute>
        <attribute name="realmName">geronimo-properties-realm</attribute>       <!-- Security Realm name -->
        <reference name="LoginService">                                 <!--reference to the login service
GBean -->
            <name>JaasLoginService</name>
        </reference>
    </GBean>

</configuration>
```

Back to Top

Configuring Security Realm

The only implementation of the `org.apache.geronimo.security.realm.SecurityRealm` interface in Geronimo is the `org.apache.geronimo.security.realm.GenericSecurityRealm` class.

`org.apache.geronimo.security.realm.GenericSecurityRealm` implements 2 interfaces: `SecurityRealm` and `ConfigurationEntryFactory`. The `GenericSecurityRealm` name is also the name of the `ConfigurationEntryFactory` implementation. That is why you can use `GenericSecurityRealm` name from your application as **application configuration entry name** passed to the `LoginContext()` constructor, see the Geronimo and JAAS section.

You need to give a name to the `GenericSecurityRealm` and configure it's authentication policy by wiring up login modules into the realm. Login modules are not wired up by themselves but are **qualified by their use** in the computation of authentication outcome.

The list of login modules that must be configured into the `GenericSecurityRealm` is specified with the `org.apache.geronimo.security.jaas.JaasLoginModuleUse` GBean. It is injected with the `LoginModuleGBean`, the value of the control-flag that specifies how authentication outcome of this login module must be combined with the authentication outcomes of other login modules to compute authentication result, and a reference to the next `LoginModuleUse` definition.

You may wonder why do you need a linked list of GBeans. Wouldn't it be much easier to list parameters for the `GenericSecurityRealm` and be done with it?

The answer is that Geronimo is an **IOC container** and one of it's major functions in addition to dependency injection is dependency management. That means that if GBean A depends on GBean B, then GBean B will be started by the Geronimo container before GBean A. Login modules are deployed as GBeans and `GenericSecurityRealm` GBean depends on the login module GBeans. If you just list login module object names together with the control flags, Geronimo container would not be able to resolve this dependencies and you would not have a guarantee that all login modules wired up into the generic Security Realm are up and running before `GenericSecurityRealm` comes online.

But still, an effort required to configure login modules into `GenericSecurityRealm` with the list of `LoginModuleUse` GBeans may seem excessive. To help with this kind of problems GBean definition syntax allows for *syntactic sugar* in the form of **xml-reference** element. At this point it is necessary to emphasize that this is just a reference that gets processed at the deployment time to create and wire up all GBeans that otherwise would have been explicitly defined. We will show the use of **xml-reference** in the `GenericSecurityRealm` configuration later.

Here is an example of generic-security-realm setup, we want to wire the `GenericSecurityRealm` named **geronimo-properties-realm** with the login module
named **properties-login** that authenticates against a property file. Our **Security Realm** authentication policy requires **properties-login** module authentication to succeed.

```
<GBean name="geronimo-properties-realm" class="org.apache.geronimo.security.realm.GenericSecurityRealm">

   <!-- security-realm name; this is a name of the Security Realm as well as the name of
     -- the configuration entry used by the application -->

   <attribute name="realmName">geronimo-properties-realm</attribute>

   <!-- reference to the head of the login module use list -->
   <reference name="LoginModuleConfiguration">
      <name>properties-login</name>
   </reference>

   <!-- server-info reference is passed to most GBeans -->
   <reference name="ServerInfo">
      <module>org/apache/geronimo/System</module><name>ServerInfo</name>
   </reference>

   <!-- reference to the login-service GBean -->
   <reference name="LoginService"><name>JaasLoginService</name></reference>
</GBean>

<!-- this is the head of the login-module-use list -->
<GBean name="properties-login" class="org.apache.geronimo.security.jaas.JaasLoginModuleUse">

   <!-- login module must succeed -->
   <attribute name="controlFlag">REQUIRED</attribute>

   <!-- reference to the login module -->
   <reference name="LoginModule">
         <name>properties-login</name>
   </reference>
</GBean>

<!-- this is login module GBean -->
<GBean name="properties-login" class="org.apache.geronimo.security.jaas.LoginModuleGBean">
   <attribute name="loginModuleClass">
      org.apache.geronimo.security.realm.providers.PropertiesFileLoginModule
   </attribute>
   <attribute name="serverSide">true</attribute>

   <!-- login module specific options -->
   <attribute name="options">
      usersURI=var/security/users.properties        <!-- user database -->
      groupsURI=var/security/groups.properties        <!-- group database -->
   </attribute>
   <attribute name="loginDomainName">geronimo-properties</attribute>
</GBean>
```

It does not look too bad in this example but imagine that you have 2 login modules in the Security Realm and how many GBean dependencies you have to configure.

Note that the order in which all these elements are defined does not matter. If you look at the deployment plans, you will find that login-module GBeans are defined first (as they represent elements of reuse by the `GenericSecurityRealm` GBeans). `GenericSecurityRealm` GBeans and `JaasLoginModul eUse` GBeans are normally close to each other.

Back to Top

Configuring GenericSecurityRealm using xml-reference

The reason for the introduction of the **xml-reference** element in GBean syntax was explained earlier. But just to repeat: it is a *syntactic sugar* that allows **pr oblem friendly** xml syntax in GBean definition.

Problem-friendly xml syntax for the login module configuration is defined by the "http://geronimo.apache.org/xml/ns/loginconfig-1.0" xml namespace.

The following example briefly shows how the `LoginConfig` schema is used.

```
<GBean name="geronimo-properties-realm"
   class="org.apache.geronimo.security.realm.GenericSecurityRealm">

   <!-- security-realm name; this name is reused by the
     -- configuration-entry-factory interface implementation by the
     -- generic-security-realm; you may use this name as application
     -- configuration name parameter passed to the LoginContext constructor -->

   <attribute name="realmName">geronimo-properties-realm</attribute>

   <!-- xml reference, better than before? -->
   <xml-reference name="LoginModuleConfiguration">
      <lc:login-config xmlns:lc="http://geronimo.apache.org/xml/ns/loginconfig">
         <lc:login-module control-flag="REQUIRED" server-side="true">
            <lc:login-domain-name>client-properties-realm</lc:login-domain-name>
            <lc:login-module-class>
                org.apache.geronimo.security.realm.providers.PropertiesFileLoginModule
            </lc:login-module-class>
            <lc:option name="usersURI">
               var/security/users.properties
            </lc:option>
            <lc:option name="groupsURI">
               var/security/groups.properties
            </lc:option>
         </lc:login-module>
      </lc:login-config>
   </xml-reference>
   <!-- server-info reference is passed to most GBeans -->
   <reference name="ServerInfo">
      <module>org/apache/geronimo/System</module><name>ServerInfo</name>
   </reference>

   <!-- reference to the login-service GBean -->
   <reference name="LoginService"><name>JaasLoginService</name></reference>
</GBean>
```

Back to Top

Configuring Login module

Login module is configured with `org.apache.geronimo.security.jaas.LoginModuleGBean`. It takes `loginModuleClass` attribute that specifies the login module implementation class. Other interesting parameters are options and `loginDomainName`.

The following is an example of a login module that uses property files as authentication database. Values of property files are passed as options attribute.

```
<GBean name="properties-login"
   class="org.apache.geronimo.security.jaas.LoginModuleGBean">
   <attribute name="loginModuleClass">
      org.apache.geronimo.security.realm.providers.PropertiesFileLoginModule
   </attribute>
   <attribute name="serverSide">true</attribute>
   <attribute name="options">
            usersURI=var/security/users.properties
            groupsURI=var/security/groups.properties
    </attribute>
    <attribute name="loginDomainName">geronimo-properties-realm</attribute>
</GBean>
```

Back to Top