HPCC Systems®

Security Manager Plug-in Framework

Boca Raton Documentation Team

LexisNexis® Risk Solutions

Security Manager Plug-in Framework

Boca Raton Documentation Team

We welcome your comments and feedback about this document via email to <docfeedback@hpccsystems.com> Please include **Documentation Feedback** in the subject line and reference the document name, page numbers, and current Version Number in the text of the message.

2016 Version 6.2.0-1

Security Manager Development

The Security Manager Plug-in framework provides a mechanism for the creation and deployment of custom security manager plug-ins.

Plug-in Development

A custom Security Manager Plug-in consists of a **library** (.so or .dll) file which provides implementation(s) of the iSecManager interface, a **configuration definition** (articulated as an XSD schema), and a component declaration (buildset.xml file).

Library requirements

- Must implement the ISecManager interface
- Must expose a factory method which returns instances of the ISecmanager implementation.

  Example of a standard factory method name :

```
extern "C"
{
    ISecManager * createInstance(const char *serviceName,
                                 IPropertyTree &secMgrCfg,
                                 IPropertyTree &bndCfg);
}
```

  The framework expects to have access to the "createInstance()" method, if the developer chooses to provide other factory methods, it can override the default name in configuration, but must have the expected signature:

```
ISecManager methodname(const char *, IPropertyTree &, IPropertyTree &)
```

Buildset definition - The plug-in declares itself as an HPCC Security Manager Plug-in component, and declares the location of the plug-in files and the configuration definition schema.

EXAMPLE:

```
<Environment>
   <Programs>
      <Build name="_" url="/opt/HPCCSystems">
          <BuildSet  deployable="no"
                     installSet="deploy_map.xml"
                     name="mysecuritypluginname"
                     path="componentfiles/mysecuritypluginname"
                     processName="MySecurityPluginName"
                     schema="myCustom_secmgr.xsd">
          </BuildSet>
      </Build>
   </Programs>
</Environment>
```

Configuration Definition - The plug-in must provide a definition of the configuration elements and the structure it expects to receive at the time it is instantiated. The XSD file is consumed by the HPCC Configuration manager component and is rendered as a GUI form. The configuration definition is defined as an element of the component name (as declared in the buildset) followed by attributes and/or complex elements.

There are four attributes every plug-in is required to declare in its configuration definition in addition to any custom configuration defined by the plug-in: 'type', 'name', 'libName', and 'instanceFactoryName'

- **type** - This attribute should be read-only and set to 'SecurityManager'
- **name** - The name of the custom Security Manager Plug-in instance
- **libName** - The name of the library which provides instances of this Security Manager Plug-in type
- **instanceFactoryName** - Name of the method provided by the library, which is responsible for creating instances of the Security Manager Plug-in

EXAMPLE:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="MySecurityPluginType">
     <xs:complexType>
        <xs:attribute name="type" type="SecurityManager"
                      use="required" default="SecurityManager">
           <xs:annotation><xs:appinfo>
                 <viewType>hidden</viewType>
              </xs:appinfo></xs:annotation>
        </xs:attribute>
        <xs:attribute name="name" type="xs:string" use="required">
           <xs:annotation><xs:appinfo>
                 <tooltip>Name for this Security Manager Plugin instance</tooltip>
                 <required>true</required>
              </xs:appinfo></xs:annotation>
        </xs:attribute>
        <xs:attribute name="libName" type="xs:string" use="optional">
           <xs:annotation><xs:appinfo>
                  <tooltip>The Security Manager library name (.so)</tooltip>
               </xs:appinfo></xs:annotation>
         </xs:attribute>
         <xs:attribute name="instanceFactoryName" type="xs:string"
                       use="optional" default="createInstance">
            <xs:annotation><xs:appinfo>
                  <tooltip>The factory method name in the
                   Security Mangager library (.so)</tooltip>
              </xs:appinfo></xs:annotation>
         </xs:attribute>
           <xs:sequence>
             <xs:element name="compoundOption" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                   <xs:attribute name="Option" type="xs:string" use="required">
                       <xs:annotation><xs:appinfo>
                             <tooltip>This is an example compound option element
                             which Security Manager Plug-ins can define</tooltip>
                           </xs:appinfo></xs:annotation>
                    </xs:attribute>
                </xs:complexType>
              </xs:element>
         </xs:sequence>
        <xs:attribute name="optionalAttribute" type="xs:string" use="optional">
            <xs:annotation><xs:appinfo>
                   <tooltip>This is an example optional attribute
                           which Security Manager Plug-ins can define</tooltip>
                </xs:appinfo></xs:annotation>
         </xs:attribute>
         <xs:attribute name="samplepasswordfile" type="xs:string" use="required">
            <xs:annotation><xs:appinfo>
                  <tooltip>An attribute which defines a file name required
                          by this Security Manager Plug-in</tooltip>
              </xs:appinfo></xs:annotation>
         </xs:attribute>
     </xs:complexType>
  </xs:element>
</xs:schema>
```

genenvrules.conf - (optional) This file allows the plug-in to add itself to the "do_not(automatically)_generate" list. While this is an optional file, more often than not it is actually needed.

EXAMPLE:

```
do_not_generate=mysecuritypluginname
```

Configuration transformation rules - (optional) specified as an xsl template, this set of rules can be applied to the configuration XML. Refer to XSL templates in the HPCC source tree.

Concrete Example

The HPCC Platform includes a security manager plug-in implementation (HTPasswd) and can be used as a guidance for the plug-in development process:

https://github.com/hpcc-systems/HPCC-Platform/tree/master/system/security/plugins/htpasswdSecurity

Configure and Deploy the Security Manager Plug-in

The following sections detail the process of configuring your HPCC system to use the Security Manager Plug-in.

How to Configure a Security Manager Plug-in

Once the plug-in has been installed, the plug-in can be configured onto the HPCC platform using Configuration Manager.

1. Stop all HPCC Components.

   Verify that they are stopped. You can use a single command, such as :

   ```
   sudo /opt/HPCCSystems/sbin/hpcc-run.sh -a hpcc-init status
   ```

2. Connect your web browser to the Configuration Manager web interface.

   Use the url http://<*configmgr_IP_Address*>:8015

   where <*configmgr_IP_Address*> is the IP address of the node running Configuration Manager.

3. Select the **Advanced View** radio button.

4. Select the appropriate XML configuration file.

   Note: Configuration Manager never works on the active configuration file. After you finish editing you will have to copy the environment.xml to the active location and push it out to all nodes.

5. Check the Write Access box.

6. Create an instance of the Security Manager Plug-in:

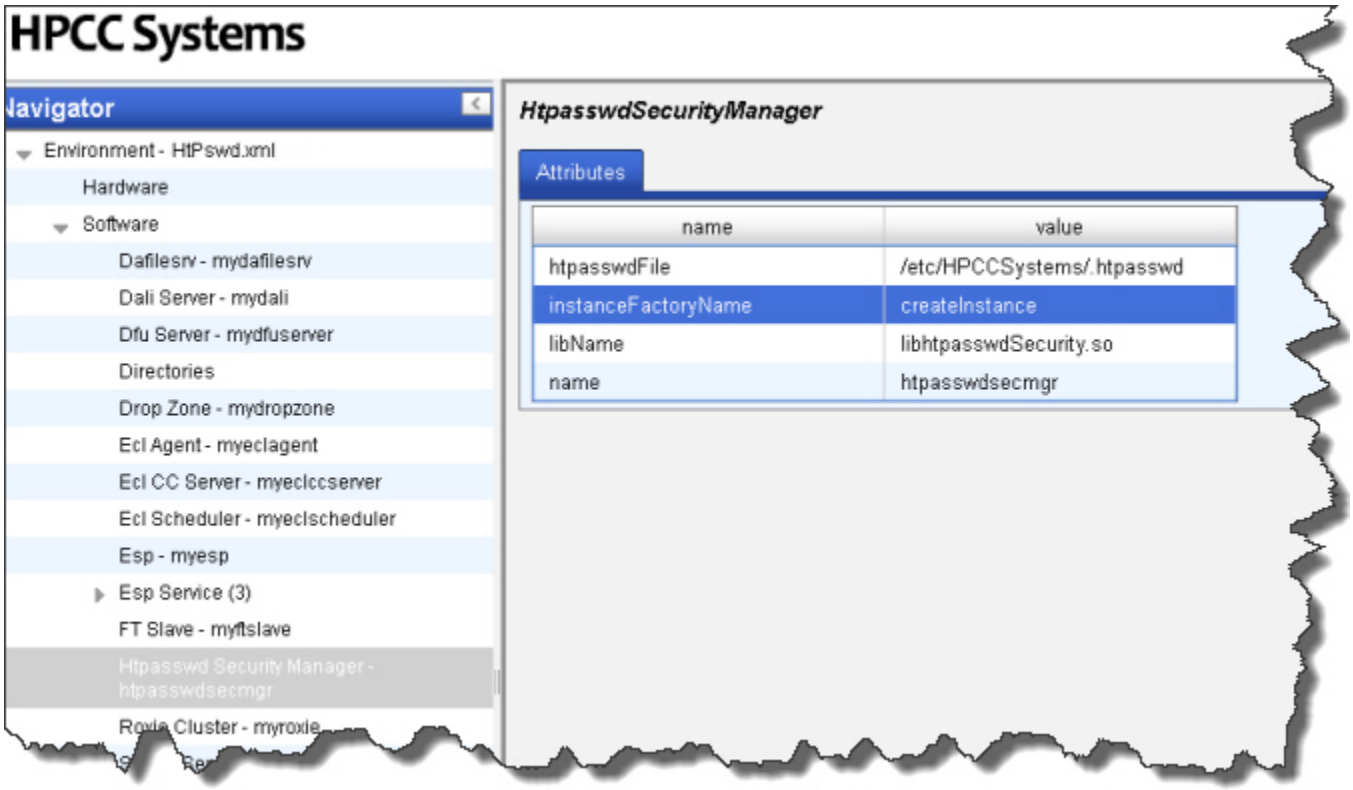
   a. Right-click on Navigator Pane on the left side.

   b. Select **New Components**

   c. Select the appropriate component <name_of_Security_Manager_plug-in>

7. Configure the Security Manager Plug-in: (Example shown using the Htpasswd plug-in*)

Figure 1. Security Mgr Configuration page



a. Provide an instance name

b. Provide a (fully qualified) library name

c. InstanceFactoryName defaults to "createInstance" if the library specified in the previous step provides an alternate factory method, it can be specified here.

d. Provide any custom entries required. In the example shown, *htpasswdFile* is a custom entry.

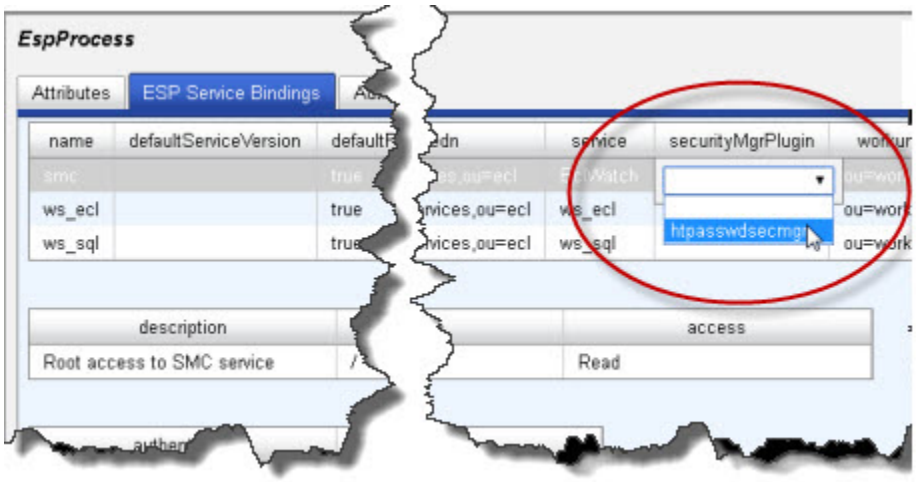
8. Associate the Security Manager Plug-in with the ESP binding(s)

   a. Click on the target Esp in the Navigator Pane on the left side.

   b. Select the **ESP** (Process) **Service bindings** tab

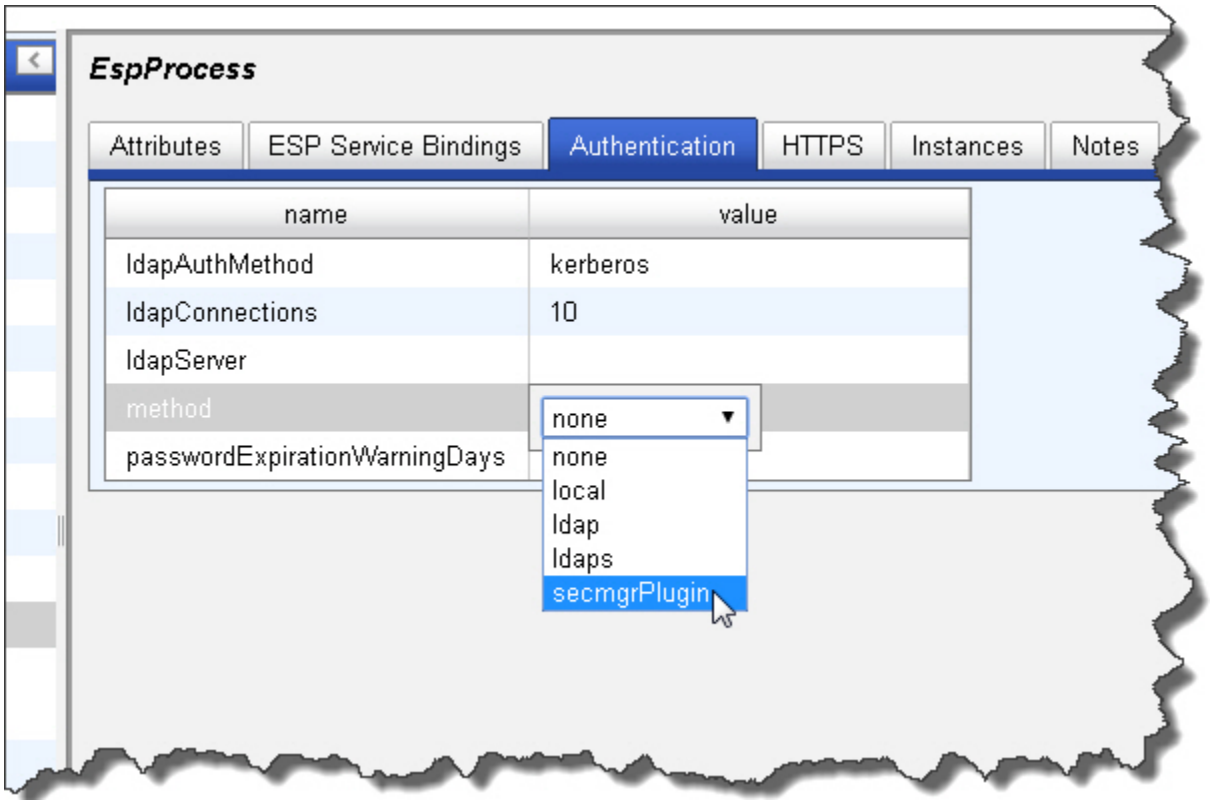
   c. On the target binding(s) select the appropriate securityMgrPlugin instance from the drop list.

Figure 2. Bind to ESP



9. Enable the use of the Security Manager Plug-in - Select the **Authentication** tab, in the method entry select **secmgrPlugin**

Figure 3. Security Mgr Configuration page



10.Save the environment file

11.Distribute the environment file to every node in your cluster

12.Restart your environment.

A video tutorial

Need further information? Check out the following video tutorial demonstrating how to configure a security plug-in.

https://www.youtube.com/watch?v=lNVwEOFkKgY&feature=youtu.be

Click the above link to watch the video.