

Chapter 6

Developing Router Initialization Scripts

The SAE manages the JUNOSe router as a client to its Common Open Policy Service (COPS) server. The SAE calls a router initialization script when the connection to the COPS client on the router is established and when the connection is terminated. The SAE uses an extension script mechanism to allow you to customize the setup of the COPS connection with the COPS client on JUNOSe routers. (See also *Router Initialization Scripts* on page 5).

This chapter contains the following sections:

- Interface Description on page 201
- Methods on page 202
- Example on page 203

Interface Description

Extension scripts interact with the SAE through an interface object called `Ssp`. The SAE exports a number of fields through the interface object to the extension script and expects the extension script to provide the entry point to the SAE.

Table 14 describes the fields that the SAE exports.

Table 14: Exported Fields

Ssp Attribute	Description
<code>Ssp.properties</code>	System properties object (class: <code>java.util.Properties</code>)—The properties should be treated as read-only by the script.
<code>Ssp.errorLog</code>	Error logger—Use the <code>Ssp.errorLog.println (message)</code> to send error messages to the log.
<code>Ssp.infoLog</code>	Info logger—Use the <code>Ssp.infoLog.println (message)</code> to send informational messages to the log.
<code>Ssp.debugLog</code>	Debug logger—Use the <code>Ssp.debugLog.println (message)</code> to send debug messages to the log.

The router initialization script must set the field `Ssp.routerInit` to a factory function that instantiates a router initialization object:

- `< VRName >` —Name of the virtual router in which the COPS client has been configured, format: `virtualRouterName@RouterName`
- `< virtualIp >` —Virtual IP address of the SAE (string, dotted decimal; for example: 192.168.254.1)
- `< realIp >` —Real IP address of the SAE (string, dotted decimal; for example, 192.168.1.20)
- `< VRip >` —IP address of the virtual router (string, dotted decimal)
- `< transportVR >` —Name of the virtual router used for routing the COPS connection, or None, if the COPS client is directly connected

The factory function must implement the following interface:

```
Ssp.routerInit(VRName,
virtualIp,
realIp,
VRip,
transportVR)
```

The factory function returns an interface object that is used to set up and tear down a connection for a given COPS server. A common case of a factory function is the constructor of a class.

The factory function is called directly after a COPS server connection is established. In case of problems, an exception should be raised that leads to the termination of the COPS connection.

Methods

Instances of the interface object must implement the following methods:

- `setup()`—Is called when the COPS server connection is established and is operational. In case of problems, an exception should be raised that leads to the termination of the COPS connection.
- `shutdown()`—Is called when the COPS server connection is terminated to the virtual router. This method should not raise any exceptions in case of problems.

Example

The following script defines a router initialization class named *SillyRouterInit*. The interface class does not implement any useful functionality. The interface class just writes messages to the infoLog when the router connection is created or terminated.

```
class SillyRouterInit:
    def __init__(self, vrName, virtualIp, realIp, vrIp, transportVr):
        """ initialize router initialization object """
        self.vrName = vrName
        Ssp.infoLog.println("SillyRouterInit created")

    def setup(self):
        """ initialize connection to router """
        Ssp.infoLog.println("Setup connection to VR %(vrName)s" %
            vars(self))

    def shutdown(self):
        """ shutdown connection to router """
        Ssp.infoLog.println("Shutdown connection to VR %(vrName)s" %
            vars(self))

#
# publish interface object to Ssp core
#
Ssp.routerInit = SillyRouterInit
```

