

Router Script Development

8

The SAE uses an extension script mechanism to allow the customized setup of the COPS connection with the COPS client.

The SAE calls the router initialization script when the connection to the COPS client on the E-series router is established and when the connection is terminated.

Refer to the *SDX Administration Guide* for more information about scripts.

Topic	Page
Interface Description	8-1
Methods	8-3
Example	8-3

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 8-1 describes the fields that are exported by the SAE.

Table 8-1 Exported fields

Ssp Attribute	Description
Ssp.properties	System properties object (class: java.util.Properties) —The properties should be treated as read-only by the script.
Ssp.errorLog	Error logger—Use the Ssp.errorLog.println (message) to send error messages to the log.
Ssp.infoLog	Info logger—Use the Ssp.infoLog.println (message) to send informational messages to the log.
Ssp.debugLog	Debug logger—Use the Ssp.debugLog.println (message) 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: virtualrouter-name@router-name
- *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 virtual router (string, dotted decimal)
- *transportVR* – name of virtual router used for routing the COPS connection, or None, if COPS client is directly connected

The factory function must implement the interface shown in Figure 8-1.

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

Figure 8-1 Interface script

The factory function returns an interface object that is used to setup and teardown 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 has been established. In case of problems, an exception should be thrown 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 thrown 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 throw any Exceptions in case of problems.

Example

Figure 8-2 shows the example script that 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
```

Figure 8-2 Example Script

