Scripting Framework

This chapter describes the Scripting framework in Cisco BAC that involves configuring java script based DPE extensions.

This chapter includes the following sections:

Overview

Scriptable extension service facilitates running the java scripts based DPE extensions. Once the extension scripts are added in the RDU database, the RDU applies the scripts to the DPE cache. When the CPE boots and attempts to establish connection with the DPE, the extension scripts are executed based on the extensions configured in the device property hierarchy.

Script's Shared Scope

Script's Shared Scope is a shared repository for extension scripts. Cisco BAC APIs like get properties from PG, fire extension specific events, debugging, logging, and so on, are put into the Script's Shared Scope so that the extension scripts can call them whenever required.

The following objects are added in the Script's Shared Scope for extension scripts:

Table 18-1 Java Objects

Object Name	API
request	CwmpHttpRequest
response	CwmpHttpResponse

Table 18-1 Java Objects

Object Name	API
session	CwmpHttpSession
scriptSupport	ScriptSupport
service	CwmpHttpService
log	DefaultLogStream

By default the following frequently used Java packages are imported into the Script's Shared Scope:

```
java.lang.*;
java.util.*;
com.cisco.provisioning.cpe.cwmp.main.*;
com.cisco.provisioning.cpe.cwmp.soap.messages.*;
com.cisco.provisioning.cpe.cwmp.soap.types.*;
com.cisco.provisioning.cpe.extensions.dpe.api.events.*;
com.cisco.provisioning.cpe.extensions.dpe.faults.*;
com.cisco.csrc.dpe.extensions.script.*;
```

You can also add any third-party libraries into the Script's Scope by adding the jar files into /dpe/lib folder. During DPE start-up, the class loader will load all the jars available in /dpe/lib folder. You can import the Java packages in the extension scripts to make use of the Java classes available in the libraries loaded.

For example, you can import the java packages using the following command:

```
importPackage (Packages.com.mylib.test);
```

Note Restart the DPE after adding the jar files.

Extension Script

Extension scripts are classified into two categories:

- DPE Extension Script—These script files are main extension scripts having the main provisioning logic.

  The following script is a sample DPE extension script:

  ```
  log.debug(" Executing sample extension...");
  var params = ["Device.DeviceInfo.SoftwareVersion"];
  log.debug(" Getting SoftwareVersion info...");
  var result = getParamValues(scriptSupport, response, params);
  log.debug(" Got response from the device ==> ");
  log.debug(result.toString());
  ```

- Helper Script—These script files are utilities which can be used by multiple main extension scripts. The helper scripts are called from both main extension scripts and other helper scripts.

  The following script is a sample Helper script:

  ```
  function getParamValues(scriptSupport, response, params)
      {
          if (params && params.length > 0)
          {
  ```

```
            // With continuation support, the script gets the
            // response the in the same line as if using
            // a synchronous interface
            // sendAndReceive() sends the give message to the
            // device and receives the response message.
            return scriptSupport.sendAndReceive(response,
                scriptSupport.newGetParameterValuesMessage(params));
        }
        return null;
    }
```

Integrating Extension Scripts in BAC

The extension scripts are added to RDU database from the admin UI.

To add a new extension script to the Cisco BAC RDU:

Step 1 Choose **Configuration** on the primary navigation bar.

Step 2 Choose **Files** on the secondary navigation bar. The View Files page appears.

Step 3 Click **Add**.

The Add Files page appears.

Step 4 Choose DPE Extension Script / Helper Script from the File Type drop-down list.

Step 5 Browse for the Source File Name.

Step 6 Add the .js file in the File Name field.

Step 7 Click **Submit.**

The View Files page appears with the new file.

The extension scripts can be viewed only after exporting the .js script file from the admin UI to your local machine. For more information on exporting extension scripts, see Exporting Files, page 17-20.

Adding Extension Points to the Device Property

Extensions can be configured in the device property hierarchy at Class of Service (CoS), Device, Group, Provisioning group and CWMP defaults.

The following extension points can be configured in the device:

- /IPDevice/extensions/soapRequestSender - The extension scripts configured are executed, when a request is received from the CPE.
- /IPDevice/extensions/soapResponseSender - The extension scripts configured are executed, when a response is sent to the CPE.
- /IPDevice/extensions/incomingEventViewer - The extension scripts configured are executed, when the event (existing configuration) is received from the CPE.
- /IPDevice/extensions/outgoingEventViewer - The extension scripts configured are executed, when the event (required configuration) is sent to the CPE.

To add scriptable extensions at CWMP Defaults level:

Step 1 Choose **Configuration** on either the primary navigation bar or main menu page.

Step 2 Choose **Defaults** from the secondary navigation bar.

The CWMP Defaults page appears. See CWMP Defaults, page 17-8 for field description.

Step 3 Configure the extension scripts for the required extensions.

For example, specify *sampleExtension.js* in the **SOAP Request Sender Extension** field. Similarly, specify the required extension scripts in the **SOAP Response Sender Extension**, **Incoming Event Viewer Extension**, and **Outgoing Event Viewer Extension** fields.

For information on adding extension scripts to the RDU database, see Integrating Extension Scripts in BAC, page 18-3.

Step 4 Click **Submit**.

The scriptable extensions are added in the DPE.

Note If the extensions are not configured at Class of Service (CoS), Device, Group, or Provisioning group level in the device property hierarchy, by default, extensions configured under CWMP defaults are executed.

Configuring Extensions for Unknown Devices

You can configure the extension list for unknown devices from the admin UI from the CWMP Defaults page. This enables the DPE to retrieve the list of default extensions and execute the extension scripts, when the unknown device establishes connection with the DPE.

To configure extensions for unknown devices:

Step 1 Add the extension script to the RDU database using DPE Extension Script file.

For more details on how to add an extension script in BAC, see Integrating Extension Scripts in BAC, page 18-3.

Step 2 Configure the extension in the CWMP Defaults page.

For more details on CWMP Defaults, see CWMP Defaults, page 17-8.

Step 3 Boot an unknown device.

Step 4 Verify that the extension script is executed in DPE.

For more details on verifying the status of the extensions loaded in DPE, see Verifying Status of the Extensions in DPE, page 18-5.

Verifying Extension Scripts Availability in DPE Cache

When the device boots, it contacts DPE and invokes the extension scripts available in DPE cache based on the extensions configured for the device.

To verify whether the extensions scripts are pushed to DPE cache from RDU:

Step 1 Log into DPE CLI.

Step 2 Verify that the extension scripts are available in the DPE cache, using the following command:

```
dpe# show files
```

The following is a sample output:

```
The list of files currently in DPE cache

filename                      size

main.js                      2161
passthroughtest.js           4106
sample-firmware-image.bin    4239368
sampleextension.js           3319
samplehelper.js              2645


DPE caching 5 external files.
Listing the first 5 files, 0 files omitted
```

Verifying Status of the Extensions in DPE

The extensions scripts are executed in DPE, based on the extension points configured in device property hierarchy.

To verify if the extensions are executed in the DPE after booting the device:

Step 1 Log into DPE CLI.

Step 2 Enable debug for DPE extensions using the command:

```
debug dpe dpe-ext
```

Step 3 Boot the device.

Step 4 Observe the dpe.log file to verify if the scripts are executed in DPE. To view DPE logs see Viewing the dpe.log File, page 21-8.

The following is a sample output:

```
[Device [00000C-1234567890] Session [149eb9f53A137fed6a9c353A80000023]. Extension
[sampleextension.js].]: [Executing script [sampleextension.js] for the firs
t time]
```

Note The result is displayed in the dpe.log file.