Using Powershell 3 to Access REST

Table of Contents

Getting Started

Starting with Release 3.0 of SBC 1000/2000 firmware, **the REST license is free of charge** and you can use it for multiple scenarios.

REST APIs are provided for developers who want to programmatically integrate the SBC 1000/2000 into their applications, and for administrators who want to script interactions with the SBC 1000/2000.

An example is scripting provisioning, which allows you to program multiple SBCs with the same parameters, instead of manually programming each one. This is often the case when you need to deploy multiple SBCs in the same county, and must create the same routing and transformation tables manually each time.

This article outlines tips for accessing REST programming through Microsoft Powershell 3.

Prerequisites

Your system must include the following:

- 3.x base licenses loaded into the SBC; this will define the REST license as “unlimited"
- SBC account with REST or Administrator access level
- Windows Management Framework 3.0: https://www.microsoft.com/en-us/download/details.aspx?id=34595 (Windows 7 download: Windows6.1-KB2506143-x64.msu)

Verify the Version

Verify the version of the host using the following command.

```
$host.version
```

Log into REST

Login without certificate

```
### Allow self Sign Cert
add-type @"
    using System.Net;
    using System.Security.Cryptography.X509Certificates;

    public class IDontCarePolicy : ICertificatePolicy {
        public IDontCarePolicy() {}
        public bool CheckValidationResult(
            ServicePoint sPoint, X509Certificate cert,
            WebRequest wRequest, int certProb) {
            return true;
        }
    }
"@
[System.Net.ServicePointManager]::CertificatePolicy = new-object IDontCarePolicy

### Login
$BodyValue = "Username=restuser&Password=restpass"

$url = "https://134.56.226.215/rest/login"
Invoke-RestMethod -Uri $url -Method Post -Body $BodyValue -SessionVariable ps
```

Basic REST requests - Samples

Create resources (PUT)

```
$BodyValue = "Description=Default SIP Server"
$url = "https://134.56.226.215/rest/sipservertable/8"
Invoke-RestMethod -Uri $url -Method PUT -Body $BodyValue -WebSession $ps
```

Modify resources (POST)

```
$BodyValue = "Description=Default SIP TOTO"
$url = "https://134.56.226.215/rest/sipservertable/8"
Invoke-RestMethod -Uri $url -Method POST -Body $BodyValue -WebSession $ps
```

Get resources (GET)

```
$url = "https://134.56.226.215/rest/sipservertable/8"
Invoke-RestMethod -Uri $url -Method GET -WebSession $ps
```

Delete resources (DELETE)

```
$url = "https://134.56.226.215/rest/sipservertable/8"
Invoke-RestMethod -Uri $url -Method DELETE -Body "" -WebSession $ps
```

Advanced REST Requests

Backup Configuration

```
$DestFile = "C:\Users\plessisa\Desktop\Rest\backup.tar.gz"
$url = "https://134.56.226.215/rest/system?action=backup"
Invoke-RestMethod -Uri $url -Method POST -Body "" -WebSession $ps -OutFile $DestFile
```

Restore Configuration

```
$url= "https://134.56.226.215/rest/system?action=importconfig"
$body = "--MyBoundary`n"
$body += 'Content-Disposition: form-data; name="Filename"; filename="symphonyconfig.xml"'
$body += "`nContent-Type: application/xml`n`n"
$body += $(get-content symphonyconfig.xml -raw)
$body += "`n--MyBoundary--"

Invoke-RestMethod -uri $url -method POST -ContentType "multipart/form-data; boundary=MyBoundary" -body $body -
WebSession $ps
```

Log in with cURL

This operation allows login to the SBC with cURL, and stores the PHP Session ID into Powershell for reuse later.

```
$login = ./curl.exe -k --data "Username=restuser&Password=restpass" -i -v https://192.168.123.53/rest/login
$PHPSESSID = 0
foreach ($line in $login) {
    if ($line.contains("PHPSESSID=")) {
        $correctline = $line.split(";")
        $splitedline = $correctline.split("=")
        $PHPSESSID = $splitedline[1]
    }
}
```

Update SbcComms

This operation allows installation of Ribbon SBC Communication service updates on the ASM. Refer to Resource - sbaconfig.

Query

```
curl.exe --cookie PHPSESSID=$PHPSESSID -k -i https://134.56.226.215/rest/sbaconfig?action=sbaupgrade -F
sbaInstallFilename=@setup.msi
```

Response

```
<?xml version="1.0"?>
<root>
 <status>
  <http_code>200</http_code>
 </status>
</root>
```

Action end on status

```
$url = "https://134.56.226.215/rest/sbaactionstatus"
Invoke-RestMethod -Uri $url -Method GET -WebSession $MyPs

<?xml version="1.0"?>
<root>
 <status>
  <http_code>200</http_code>
 </status>
 <sbaactionstatus href="https://134.56.226.215/rest/sbaactionstatus">
  <rt_Success>1</rt_Success>
  <rt_ActionType>15</rt_ActionType>
  <rt_PercentComplete>100</rt_PercentComplete>
  <rt_Duration>5</rt_Duration>
  <rt_StatusText>setup.msi: Update Complete</rt_StatusText>
  <rt_AllActionsBitmap>0</rt_AllActionsBitmap>
  <rt_ActionStartTime>0</rt_ActionStartTime>
 </sbaactionstatus>
</root>
```

Basic Handler (PUT, POST, GET, DELETE)

```
Function BasicHandler {
    Param($ResultIn)
    [xml]$XmlResultIn = $ResultIn.Substring(5)
    if($XmlResultIn.root.status.http_code.contains("200")) {
        return "Success"
    } else {
        return "Error Code:<"+$XmlResultIn.root.status.app_status.app_status_entry.code+"> Param:<"+$XmlResultIn.
root.status.app_status.app_status_entry.params+">"
    }
}

####### Expected
PS C:> BasicHandler $Result
Success
or
Error Code:<20013> Param:<8|SIPServerTable>
```

Content Handler (GET)

```
[xml]$FinalResult= $Result.Substring(5)

####### Expected Result
PS C:> $FinalResult.root.sipservertable.Description
Default SIP TOTO
```