iControl Apps #22 - System Software Status

Joe Pruitt, 2010-04-01

Through the iControl interfaces, we've exposed many of the features on the BIG-IP. In this example, I'll illustrate the new System Software Status method in version BIG-IP version 10.1. This information can be found on the "System.Software Management.Image List" menu item on the BIG-IP management GUI. From this information you can determine the installed images on the BIG-IP as well as their active state and status of live software installs.

Usage

The arguments for this application are the bigip address and a username/hostname for the connection.

```
param (
  $bigip = $null,
  $uid = $null,
  $pwd = $null
)

function Show-Usage()
{
  Write-Host "Usage: SystemSoftwareStatus.ps1 host uid pwd";
  exit;
}
```

Initialization

The initialization component of this script will check for the required parameters, attempt to load the iControl Snapin into the current PowerShell runspace, and then call the local Get-SystemSWStatus function that queries the system software status information.

```
function Do-Initialize()
{
  if ( (Get-PSSnapin | Where-Object { $_.Name -eq "iControlSnapIn"}) -eq $null )
  {
    Add-PSSnapIn iControlSnapIn
  }
  $success = Initialize-F5.iControl -HostName $bigip -Username $uid -Password $pwd;

  return $success;
}

if ( ($bigip -eq $null) -or ($uid -eq $null) -or ($pwd -eq $null) )
{
  Show-Usage;
}

if ( Do-Initialize )
{
  Get-SystemSWStatus
}
else
{
  Write-Error "ERROR: iControl subsystem not initialized"
}
```

Querying Software Status

The heart of this script lies around the get_all_software_status() method in the System.SoftwareManagement interface. There is also a get_software_status() method that allows you to pass in a list of installation IDs, but we'll make it simple by querying the entire list.

The method returns a structure containing the installation id (chassis slot id and install volume) along with the product, version, build base build, active status, edition, and live install status.

Member	Description
installation_id	The blade location and hard drive slot the installation is targeted for.
product	The installed product (ie. BIG-IP)
version	The version of the product (ie. 10.1.0)
build	The build number installed (ie. 3327.0)
base_build	The base build (used for hotfixes)
active	Whether the boot location is active.
edition	The edition for the given software install (used for hotfixes).
status	The status of the live install. (ie. "none", "audited", "retry", "upgrade needed", "waiting for image", installing nn.mmm pct", "complete", "cancelling", "cancelled", or "failed"

```
function Get-SystemSWStatus()
{
  $SoftwareStatusA = (Get-F5.iControl).SystemSoftwareManagement.get_all_software_status();

  "================================================"
  "               SYSTEM SOFTWARE STATUS           ";
  "------------------------------------------------"
  "{0,10} : {1}" -f "Host", $bigip;

  foreach ($SoftwareStatus in $SoftwareStatusA)
  {
    "----------------------------------------------"
    $installation_id = $SoftwareStatus.installation_id;
    Write-Entry "Chassis Slot Id" $installation_id.chassis_slot_id;
    Write-Entry "Install Volume" $installation_id.install_volume;
    Write-Entry "Product" $SoftwareStatus.product;
    Write-Entry "Version" $SoftwareStatus.version;
    Write-Entry "Build" $SoftwareStatus.build;
    Write-Entry "Base Build" $SoftwareStatus.base_build;
    Write-Entry "Active" $SoftwareStatus.active;
    Write-Entry "Edition" $SoftwareStatus.edition;
    Write-Entry "Status" $SoftwareStatus.status;
    "----------------------------------------------"
  }
  "================================================"
}
```

Utilitiy Functions

To help with formatting, I included the Write-Entry utility function to help space the output into columns.

```
function Write-Entry()
{
  param([string]$name,
    [string]$value,
```

```
      [int]$width = 15);
   if ( $name.Length -gt 0 )
   {
      $fmt = "{0,$width}";
      if ( $value.Length -gt 0 )
      {
         $fmt += " : {1}";
      }
   }
   $fmt -f $name, $value;
}
```

Usage

text

```
PS C:\> .\SystemSoftwareStatus.ps1 bigip user pass


==============================================
            SYSTEM SOFTWARE STATUS
--------------------------------------------------
      Host : theboss
--------------------------------------------------
Chassis Slot Id : 0
 Install Volume : HD1.2
        Product : BIG-IP
        Version : 10.1.0
          Build : 3341.0
     Base Build : 3341.0
         Active : True
        Edition
         Status : complete
--------------------------------------------------
--------------------------------------------------
Chassis Slot Id : 0
 Install Volume : HD1.1
        Product : BIG-IP
        Version : 10.1.0
          Build : 3327.0
     Base Build : 3327.0
         Active : False
        Edition
         Status : complete
--------------------------------------------------
--------------------------------------------------
Chassis Slot Id : 0
 Install Volume : CF1.1
        Product : BIG-IP
        Version : 9.2.0
          Build : 119.0
     Base Build
         Active : False
        Edition
         Status : complete
--------------------------------------------------
==============================================
```

You'll see for my system, I have three installations. The active installation is running on install volumn HD1.2 with version 10.1.0, version 3341.0. I've also got an older version of 10.1 on installation ID HD1.1 and an ancient v 9.2 on my CF1.1 volume.

You can see the full script under

PowerShellSoftwareStatus

in the iControl wiki.