iControl Apps - #20 - Server Control

Joe Pruitt, 2009-22-07

The issue of server state continues to come up in the DevCentral iControl forums so I figured I'd write an application that illustrates how to get and set the state of a server object and in doing so show how the iControl API methods map to the three way toggle state of enabled, disabled, forced offline in the BIG-IP admin GUI.

Usage

The arguments for this application are the BIG-IP address, username and password along with optional arguments for the object identifier and the toggle state. I've created a few hash tables to help with converting from the state parameter's values of "enabled", "disabled", and "offline" to the appropriate monitor and session state values.

```
param (
  $bigip = $null,
  $user = $null,
  $pass = $null,
  $object = $null,
  $state = $null
);

Set-PSDebug -strict;

$MONITOR_STATE_HASH = @{};
$MONITOR_STATE_HASH.Add("enabled", "STATE_ENABLED");
$MONITOR_STATE_HASH.Add("disabled", "STATE_ENABLED");
$MONITOR_STATE_HASH.Add("offline", "STATE_DISABLED");

$SESSION_STATE_HASH = @{};
$SESSION_STATE_HASH.Add("enabled", "STATE_ENABLED");
$SESSION_STATE_HASH.Add("disabled", "STATE_DISABLED");
$SESSION_STATE_HASH.Add("offline", "STATE_DISABLED");

#--------------------------------------------------------------------------
# function Write-Usage
#--------------------------------------------------------------------------
function Write-Usage()
{
  Write-Host "Usage: ServerControl.ps1 host uid pwd [object [state]]";
  Write-Host "   object: address (Node) | address:port (Pool Member)";
  Write-Host "    state: enabled | disabled | offline";
  exit;
}
```

Initialization

The main app logic checks for whether the required BIG-IP connections parameters are supplied. If they are, it performs the initialization to load the iControl PowerShell SnapIn into the current Runspace so that the iControl calls are available to the script.

At this point if the object parameter was not supplied, it will call the local Get-ObjectState function with null as an argument. This will return results for all objects on the system. If a object is supplied but a state is not, then the Get-ObjectState function will be called for that object. Finally, if both an object and state are supplied, the script will attempt to set the object's state to the specified value.

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

  return $success;
}

if ( ($bigip -eq $null) -or ($user -eq $null) -or ($pass -eq $null) )
{
  Write-Usage;
}

if ( Do-Initialize )
{
  if ( ! $object )
  {
    # List all Node Addresses and Pool Members
    Get-ObjectState -objects $null;
  }
  elseif ( ! $state )
  {
    # List toggle state of provided object
    Get-ObjectState -objects (,$object);
  }
  else
  {
    if ( ($state -eq "enabled") -or ($state -eq "disabled") -or ($state -eq "offline") )
    {
      # Set the specified object's toggle state
      Set-ObjectState -object $object -state $state;
    }
    else
    {
        Write-Usage;
    }
  }
}
else
{
  Write-Error "ERROR: iControl subsystem not initialized"
}
```

Determining The Toggle State

When querying the Toggle state (enabled, disabled, or offline), one must look at the monitor and session enabled states. The three possible combinations of these two values will yield the appropriate toggle states.

Monitor State	Session Enabled State	Toggle State
STATE_ENABLED	STATE_ENABLED	Enabled (All traffic allowed)
STATE_ENABLED	STATE_DISABLED	Disabled (Only persistent or active connections allowed)
STATE_DISABLED	STATE_DISABLED	Forced Offline (Only active connections allowed)

The local Get-ToggleState function will take as input the monitor and session enabled state of an object and return the toggle state.

```
function Get-ToggleState()
{
  param(
    [string]$monitor_state = $null,
    [string]$session_enabled_state = $null
  );
  $state = $null;
  if ( $monitor_state -and $session_enabled_state )
  {
    if ( ($monitor_state -eq "STATE_ENABLED") -and ($session_enabled_state -eq "STATE_ENABLED") )
    {
      $state = "enabled";
    }
    elseif ( ($monitor_state -eq "STATE_ENABLED") -and ($session_enabled_state -eq "STATE_DISABLED")
    {
      $state = "disabled";
    }
    elseif ( ($monitor_state -eq "STATE_DISABLED") -and ($session_enabled_state -eq "STATE_DISABLED")
    {
      $state = "offline";
    }
  }
  $state;
}
```

Querying The Server State

The application was designed to allow either an address (node address) or address:port (pool member) for the input object. The Get-ObjectState function will determine whether we are looking for a node address or pool member and call the appropriate method depending on the object type.

The Get-NodeAddressState function will call the LocalLB.NodeAddress.get_monitor_status() and LocalLB.NodeAddresss.get_session_enabled_state() functions for the specified node address and display the determined toggle status value.

The Get-PoolMemberState function is a bit more complex because it needs to do a reverse lookup for all pools that have the specified pool member in it's configuration. When a match is found, the pool member along with it's associated pool and status are displayed.

```
function Get-ObjectState()
{
  param([string[]]$objects = $null);

  if ( ! $objects )
  {
    $objects = Get-AllObjects;
  }
  foreach ($object in $objects)
  {
    $tokens = $object.Split((, ":"));
    if ( $tokens.Length -eq 1 )
    {
      # Node Address
      Get-NodeAddressState -address $tokens[0];
    }
    elseif ( $tokens.Length -eq 2 )
    {
```

```
    {
      # Pool Member
      Get-PoolMemberState -address $tokens[0] -port $tokens[1];
    }
    else
    {
      Write-Host "Invalid object '$object'";
    }
  }
}

function Get-NodeAddressState()
{
  param([string]$address = $null);

  $state = $null;
  if ( $address )
  {
    $MonitorStatusA = (Get-F5.iControl).LocalLBNodeAddress.get_monitor_status( (,$address));
    $monitor_state = "STATE_ENABLED";
    if ( $MonitorStatusA[0] -eq "MONITOR_STATUS_FORCED_DOWN" )
    {
      $monitor_state = "STATE_DISABLED";
    }

    $EnabledStateA = (Get-F5.iControl).LocalLBNodeAddress.get_session_enabled_state( (,$address));
    $session_enabled_state = $EnabledStateA[0];

    $state = Get-ToggleState -monitor_state $monitor_state -session_enabled_state $session_enabled_st
    New-ObjectStatus -object $address -state $state;
  }
}

function Get-PoolMemberState()
{
  param([string]$address = $null, [string]$port = $null);
  if ( $address -and $port )
  {
    $state = $null;
    $pool_list = (Get-F5.iControl).LocalLBPool.get_list();
    $memberSessionStateAofA = (Get-F5.iControl).LocalLBPoolMember.get_session_enabled_state($pool_lis
    $monitorStatusAofA = (Get-F5.iControl).LocalLBPoolMember.get_monitor_status($pool_list);

    for ($i=0; $i-lt$pool_list.Length; $i++)
    {
      for($j=0; $j-lt$memberSessionStateAofA[$i].Length; $j++)
      {
        $memberSessionState = $memberSessionStateAofA[$i][$j];
        $monitorStatus = $monitorStatusAofA[$i][$j];

        if ( ($monitorStatus.member.address -eq $address) -and
             ($monitorStatus.member.port -eq $port) )
        {
          # found a match
          $session_enabled_state = $memberSessionState.session_state;
          $monitor_state = "STATE_ENABLED";
          if ( $monitorStatus.monitor_status -eq "MONITOR_STATUS_FORCED_DOWN" )
          {
```

```
            $monitor_state = "STATE_DISABLED";
          }

          $state = Get-ToggleState -monitor_state $monitor_state -session_enabled_state $session_enab
          New-ObjectStatus -object "${address}:${port}" -parent "$($pool_list[$i])" -state $state;
        }
      }
    }
  }
}
```

Setting The Server State

Simliar to the Get-ObjectState function, the Set-ObjectState function will determine whether the object is a node address or pool member and call the appropriate function.

The Set-NodeAddressState function will call the LocalLB.NodeAddress.set_monitor_state() and LocalLB.NodeAddress.set_session_enabled_state() functions with the appopriate monitor and session state as described in the above table.

The Set-PoolMemberState function again will do a reverse lookup to find all containing pools and set the values accordingly for all pools that contain that pool member.

```
function Set-ObjectState()
{
  param(
    [string]$object = $null,
    [string]$state = $null
  );
  if ( $object -and $state )
  {
    $tokens = $object.Split((,":"));
    if ( $tokens.Length -eq 1 )
    {
      Set-NodeAddressState -address $tokens[0] -state $state;
    }
    elseif ( $tokens.Length -eq 2 )
    {
      Set-PoolMemberState -address $tokens[0] -port $tokens[1] -state $state;
    }
  }
}

function Set-NodeAddressState()
{
  param(
    [string]$address = $null,
    [string]$state = $null
  );

  if ( $address -and $state )
  {
    $monitor_state = $MONITOR_STATE_HASH[$state];
    $session_state = $SESSION_STATE_HASH[$state];
    if ( $monitor_state -and $session_state )
    {
      (Get-F5.iControl).LocalLBNodeAddress.set_monitor_state( (,$address), (,$monitor_state));
      (Get-F5.iControl).LocalLBNodeAddress.set_session_enabled_state( (,$address), (,$session_state))
```

```
      Get-NodeAddressState -address $address;
    }
  }
}
function Set-PoolMemberState()
{
  param(
    [string]$address = $null,
    [string]$port = $null,
    [string]$state = $null
  );
  if ( $address -and $port -and $state )
  {
    $pool_list = (Get-F5.iControl).LocalLBPool.get_list();
    $memberDefAofA = (Get-F5.iControl).LocalLBPool.get_member($pool_list);

    $monitor_state = $MONITOR_STATE_HASH[$state];
    $session_state = $SESSION_STATE_HASH[$state];

    Write-Host "state: $state; monitor_state $monitor_state; session_state: $session_state";

    for ($i=0; $i-lt$pool_list.Length; $i++)
    {
      for($j=0; $j-lt$memberDefAofA[$i].Length; $j++)
      {
        $member = $memberDefAofA[$i][$j];

        if ( ($member.address -eq $address) -and ($member.port -eq $port) )
        {
          # found a match
          $memberMonitorState = New-Object -TypeName iControl.LocalLBPoolMemberMemberMonitorState;
          $memberMonitorState.member = $member;
          $memberMonitorState.monitor_state = $monitor_state;
          (Get-F5.iControl).LocalLBPoolMember.set_monitor_state(
            (,$pool_list[$i]),
            (,(,$memberMonitorState))
          );

          $memberSessionState = New-Object -TypeName iControl.LocalLBPoolMemberMemberSessionState;
          $memberSessionState.member = $member;
          $memberSessionState.session_state = $session_state;
          (Get-F5.iControl).LocalLBPoolMember.set_session_enabled_state(
            (,$pool_list[$i]),
            (,(,$memberSessionState))
          )

          Get-PoolMemberState -address $address -port $port;
        }
      }
    }
  }
}
```

Utilitiy Functions

What's an app without a bit of utility? The New-ObjectStatus function is mainly to allow PowerShell to build a nicely formatted table of the results. An object is created with the properties of Object, Parent, and State and those property values are populated with the input to the function and the resulting object is passed out through the pipeline.

```
function New-ObjectStatus()
{
  param(
    [string]$object = $null,
    [string]$parent = $null,
    [string]$state = $null
  );
  $o = $null;
  if ( $object -and $state )
  {
    $o = 1 | select "Object", "Parent", "State";
    $o.Object = $object;
    $o.State = $state;
    if ($parent) { $o.Parent = $parent; }
  }
  $o;
}
```

Example session

foo

```
PS D:\> .\ServerControl.ps1 bigip user pass
Object                                  Parent                                  State
------                                  ------                                  -----
1.1.1.1                                                                         enabled
1.1.1.2                                                                         disabled
1.1.1.3                                                                         enabled
1.2.3.4                                                                         enabled
10.10.10.148                                                                    enabled
10.10.10.148:22                         catbert-ssh                             enabled
10.10.10.148:80                         catbert-http                            enabled
10.10.10.149                                                                    enabled
10.10.10.149:22                         xpbert-ssh                              enabled
10.10.10.149:80                         xpbert-http                             enabled
10.10.10.149:80                         pool_1                                  enabled
10.10.10.149:80                         xpbert-http                             enabled
10.10.10.149:80                         pool_1                                  enabled
10.10.10.149:81                         xpbert-http                             enabled
10.10.10.201                                                                    enabled
10.10.10.201:80                         dc-sea-web                              enabled
10.10.10.202                                                                    enabled
10.10.10.202:80                         dc-sea-web                              enabled
10.10.10.203                                                                    enabled
10.10.10.203:80                         dc-sea-media                            enabled
10.10.10.204                                                                    enabled
10.10.10.211                                                                    enabled
10.10.10.211:80                         dc-llix-web                             enabled
10.10.10.212                                                                    enabled
10.10.10.212:80                         dc-llix-web                             enabled
10.10.10.213                                                                    enabled
10.10.10.213:80                         dc-llix-media                           enabled
20.20.20.101                                                                    enabled
20.20.20.101:80                         pool_1                                  enabled
```

```
20.20.20.101:80                         pool_1                                  enabled
20.20.20.102                                                                    enabled
20.20.20.102:80                         pool_2                                  enabled
30.30.30.149                                                                    enabled

PS D:\> .\ServerControl.ps1 bigip user pass 10.10.10.149:80
Object                                  Parent                                  State
------                                  ------                                  -----
10.10.10.149:80                         xpbert-http                             enabled
10.10.10.149:80                         pool_1                                  enabled

PS D:\> .\ServerControl.ps1 bigip user pass 10.10.10.149:80 disabled
Object                                  Parent                                  State
------                                  ------                                  -----
10.10.10.149:80                         xpbert-http                             disabled
10.10.10.149:80                         pool_1                                  disabled

PS D:\> .\ServerControl.ps1 bigip user pass 10.10.10.149:80 offline
Object                                  Parent                                  State
------                                  ------                                  -----
10.10.10.149:80                         xpbert-http                             offline
10.10.10.149:80                         pool_1                                  offline

PS D:\> .\ServerControl.ps1 bigip user pass 10.10.10.149 enabled
Object                                  Parent                                  State
------                                  ------                                  -----
10.10.10.149                                                                    enabled
```

Conclusion

This appliation illustrates the logic in emulating the functionality of the three-way toggle in the BIG-IP admin GUI with the approrpriate methods in the iControl API and gives you a tool to easily enable/disable servers.

You can download the full code for this script in the iControl CodeShare under PsServerControl.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com