Java iControl Objects - Networking SelfIP

Joe Pruitt, 2010-01-12

This is the fifth article in the "Java iControl Objects" series in which I define a set of objects that implement the iControl methods in various iControl interfaces. The previous articles covered the LocalLB Rule, Pool and PoolMember as well as the System Service interfaces.

- Java iControl Objects – System Service
- Java iControl Objects – LTM Pool
- Java iControl Objects – LTM PoolMember
- Java iControl Objects – LTM Rule

For this article, I'm going to move to a different interface and look at Networking SelfIPs. Self IP addresses are a way to assign additional local addresses to the LTM. These can be used for various purposes such as providing alternate management interfaces for configuration and reporting purposes.

Prerequisites

The code in this article, relies on the iControl library for Java that can be downloaded in it's DevCentral labs project. If you haven't watched it already, check out my previous article on Configuring Eclipse for iControl development with Java.

The Class

The Self IP is defined by it's network address, netmask, floating state, unit id, and VLAN. I've included these attributes in the class definition along with accessor methods to get and set them. Once a Self IP is created, you cannot modify the address so that attribute does not have a set method associated with it.

```
 1: package iControl.Objects.Networking;
 2:
 3: import iControl.Objects.System.Service;
 4:
 5: public class SelfIP {
 6:   private iControl.Interfaces _interfaces = null;
 7:   private String _address = null;
 8:   private String _netmask = null;
 9:   private iControl.CommonEnabledState _floating_state = null;
10:   private long _unit_id = -1;
11:   private String _vlan = null;
12:
13:   //--------------------------------------------------------------------
14:   // Member Accessors
15:   //--------------------------------------------------------------------
16:   public iControl.Interfaces getInterfaces() { return _interfaces; }
17:   public void setInterfaces(iControl.Interfaces interfaces) { _interfaces = interfaces; }
18:
19:   public String getAddress() { return _address; }
20:   public void setAddress(String address) { _address = address; }
21:
22:   //--------------------------------------------------------------------
23:   // Constructors
24:   //--------------------------------------------------------------------
25:   public SelfIP(iControl.Interfaces interfaces, String address)
26:   {
27:     _interfaces = interfaces;
28:     _address = address;
29:   }
```

Member Accessors

In this class, I've added local caching of the attribute accessor methods. This way, once you query the attribute, it will only have to poll the BIG-IP once. An example of this is with the following netmask accessor. The internal _netmask holds the value and is updated only if the value has not been set yet.

```
1: public String getNetmask() throws Exception
2: {
3:   if ( null == _netmask )
4:   {
```

```
    validateMembers();
    String [] self_ips = { _address };
    String [] netmasks = _interfaces.getNetworkingSelfIP().get_netmask(self_ips);
    _netmask = netmasks[0];
  }
  return _netmask;
}
public void setNetmask(String netmask) throws Exception
{
  validateMembers();
  String [] self_ips = { _address };
  String [] netmasks = { netmask};
  _interfaces.getNetworkingSelfIP().set_netmask(self_ips, netmasks);
  _netmask = netmask;
}
```

Public Methods

I've included 3 public methods in this class.

- create – this method will create the Self IP if it doesn't already exist.
- remove – This method will remove the Self IP from the configuration on the BIG-IP.
- syncProperties – This method queries all the object attributes and loads them into the local cache.

```
public void create(
  String vlan,
  String netmask,
  long unit_id,
  iControl.CommonEnabledState state
) throws Exception
{
  validateMembers();

  String [] self_ips = { _address };
  String [] vlan_names = { vlan };
  String [] netmasks = { netmask };
  long [] unit_ids = { unit_id };
  iControl.CommonEnabledState [] states = { state };

  _interfaces.getNetworkingSelfIP().create(self_ips, vlan_names, netmasks, unit_ids, states);

  _unit_id = unit_id;
  _netmask = netmask;
  _floating_state = state;
  _vlan = vlan;
}

public void remove() throws Exception
{
  validateMembers();
  String [] self_ips = { _address };
  _interfaces.getNetworkingSelfIP().delete_self_ip(self_ips);
}

public void syncProperties() throws Exception
{
  validateMembers();
  getFloatingState();
  getNetmask();
  getUnitId();
  getVLAN();
}
```

Static Methods

As with the classes in my previous articles, I've included the getList() static method. In here, I've added ones for create() and remove_all() as well. The getList() method will return an array of SelfIP objects loaded with all their attributes. The create() method does just what it's name implies. The remove_all() method should be used with care as it removes ALL Self IP objects on the BIG-IP. There's no undo, so use at your own risk.

```
public static SelfIP [] getList(iControl.Interfaces interfaces) throws Exception
{
  SelfIP [] selfips = null;
  if ( null != interfaces )
  {
    String [] selfip_list = interfaces.getNetworkingSelfIP().get_list();

    selfips = new SelfIP[selfip_list.length];
    for(int i=0; i<selfip_list.length; i++)
    {
      selfips[i] = new SelfIP(interfaces, selfip_list[i]);
```

```
    }
  }
  return selfips;
}

public static SelfIP create
(
  iControl.Interfaces interfaces,
  String address,
  String vlan,
  String netmask,
  long unit_id,
  iControl.CommonEnabledState state
) throws Exception
{
  SelfIP selfIP = null;
  if ( (null != interfaces) )
  {
    selfIP = new SelfIP(interfaces, address);
    selfIP.create(vlan, netmask, unit_id, state);
  }
  return selfIP;
}

public static void remove_all(iControl.Interfaces interfaces) throws Exception
{
  if ( (null != interfaces) )
  {
    interfaces.getNetworkingSelfIP().delete_all_self_ips();
  }
}
```

Example Code

The following example function will first query the list of Self IPs and print to the console their settings. It will then create a new Self IP Address. It then queries the list again to show that it was created. After that, the SelfIP object is deleted and finally the list is queried again to show that the task completed successfully.

```
public void testSelfIp(String[] args)
{
  try
  {
    iControl.Interfaces interfaces = new iControl.Interfaces();
    interfaces.initialize(args[0], args[1], args[2]);

    // Get list of existing SelfIPs
    iControl.Objects.Networking.SelfIP [] selfips =
      iControl.Objects.Networking.SelfIP.getList(interfaces);
    System.out.println("Existing SelfIP List");
    System.out.println("--------------------");
    for(int i=0; i<selfips.length; i++)
    {
      selfips[i].syncProperties();

      System.out.println("[" + selfips[i].getUnitId() + "] " +
        selfips[i].getAddress() + "/" + selfips[i].getNetmask() +
        ", VLAN='" + selfips[i].getVLAN() +
        "', FSTATE='" + selfips[i].getFloatingState() + "'"  );
    }

    // Create SelfIP
    System.out.println("Creating SelfIP 20.20.20.1...");
    iControl.Objects.Networking.SelfIP.create(interfaces, "20.20.20.1",
      "external1", "255.255.255.0", 0, iControl.CommonEnabledState.STATE_DISABLED);
    System.out.println("SelfIP 20.20.20.1 successfully created...");

    iControl.Objects.Networking.SelfIP selfip2 =
      new iControl.Objects.Networking.SelfIP(interfaces, "20.20.20.1");

    selfip2.syncProperties();
    System.out.println("[" + selfip2.getUnitId() + "] " +
        selfip2.getAddress() + "/" + selfip2.getNetmask() +
        ", VLAN='" + selfip2.getVLAN() +
        "', FSTATE='" + selfip2.getFloatingState() + "'"  );

    // Get list of existing SelfIPs
    selfips = iControl.Objects.Networking.SelfIP.getList(interfaces);
    System.out.println("Existing SelfIP List");
    System.out.println("--------------------");
    for(int i=0; i<selfips.length; i++)
    {
      selfips[i].syncProperties();

      System.out.println("[" + selfips[i].getUnitId() + "] " +
```

```
47:         selfips[i].getAddress() + "/" + selfips[i].getNetmask() +
48:         ", VLAN='" + selfips[i].getVLAN() +
49:         "', FSTATE='" + selfips[i].getFloatingState() + "'"  );
50:     }
51:
52:     // Remove SelfIP
53:     System.out.println("Removing SelfIP 20.20.20.1...");
54:     selfip2.remove();
55:     System.out.println("SelfIP 20.20.20.1 successfully removed...");
56:
57:     // Get list of existing SelfIPs
58:     selfips = iControl.Objects.Networking.SelfIP.getList(interfaces);
59:     System.out.println("Existing SelfIP List");
60:     System.out.println("--------------------");
61:     for(int i=0; i<selfips.length; i++)
62:     {
63:       selfips[i].syncProperties();
64:
65:       System.out.println("[" + selfips[i].getUnitId() + "] " +
66:         selfips[i].getAddress() + "/" + selfips[i].getNetmask() +
67:         ", VLAN='" + selfips[i].getVLAN() +
68:         "', FSTATE='" + selfips[i].getFloatingState() + "'"  );
69:     }
70:
71:   }
72:   catch(Exception ex)
73:   {
74:     ex.printStackTrace(System.out);
75:   }
76: }
```

View The Source

The source code can be found in the iControl CodeShare under JavaObjectNetworkingSelfip.

Related Articles on DevCentral

- iControl 101 #23 - Module Resource Provisioning > DevCentral > F5 ...
- iControlsh - An iControl Based Console BIG-IP Shell > DevCentral ...
- Java iControl Objects - System Service > DevCentral > F5 ...
- F5 DevCentral > Community > Group Details - iControl Assembly
- What is iControl? > DevCentral > F5 DevCentral > Tech Tips
- F5 DevCentral > Community > Group Details - Python iControl Library
- Ruby iControl Wrapper
- Ruby meets iControl: Creating VIPs > DevCentral > F5 DevCentral ...

Technorati Tags: selfip, networking, iControl, Joe Pruitt

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