xConnect BigIP Configuration and Testing for HA

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Rev | Date | Author | Description |
| 1.0 | 11/9/2012 | Michael Lampi | Preliminary Release Version |
| 1.1 | 11/12/2012 | Michael Lampi | Add Visio diagrams |
| 1.2 | 11/13/2012 | Michael Lampi | Change 8090 monitor from gateway\_icmp to http |
| 1.3 | 11/19/2012 | Michael Lampi | Add configuration verification |
| 1.4 | 11/21/2012 | Michael Lampi | Add test framework |
| 1.5 | 1/15/2013 | Michael Lampi | Modify monitors to use new endpoint, change timeout values |
|  | 1/22/2013 | Michael Lampi | Change Action on Service Down to Reselect |
|  | 2/6/2013 | Michael Lampi | Correct Match Across Virtual Servers setting, both in text and screen shot. |

**Document Approvers & Sign-Off**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Approver | Role | Document Accept/Reject |
| [Date] | [Insert Approver’s Name] | [Insert role] | [Accept/Reject] |

**Table of Contents**

[1 Introduction 5](#_Toc346030997)

[1.1 Definitions 5](#_Toc346030998)

[2 xConnect Communications 6](#_Toc346030999)

[2.1 xConnect Authentication Subsystem 6](#_Toc346031000)

[3 Configuring F5 BigIP for HA 10](#_Toc346031001)

[3.1 Overall HA Configuration 10](#_Toc346031002)

[3.2 Setting Up Monitors 12](#_Toc346031003)

[3.3 Setting Up Nodes 16](#_Toc346031004)

[3.4 Setting Up a Pool 18](#_Toc346031005)

[3.5 Defining HA iRules 24](#_Toc346031006)

[3.5.1 JSessionPersistence 26](#_Toc346031007)

[3.5.2 SingleNodePersistence 27](#_Toc346031008)

[3.6 Profiles 28](#_Toc346031009)

[3.7 Setting up a Virtual Server 30](#_Toc346031010)

[3.7.1 Buzz-parks-net-8080 30](#_Toc346031011)

[3.7.2 Buzz-parks-net-8090 Virtual Server 33](#_Toc346031012)

[3.7.3 Sit-idms-disc-server Virtual Server 36](#_Toc346031013)

[3.8 BigIP Self IPs 40](#_Toc346031014)

[4 Testing F5 BigIP for HA 41](#_Toc346031015)

[4.1 Verify Basic Functionality 41](#_Toc346031016)

[4.2 Verify Pool Monitors Are Correct 42](#_Toc346031017)

[4.3 HA Tests - XBRCs 43](#_Toc346031018)

[4.3.2 Master to Slave Failover 43](#_Toc346031019)

[4.3.3 Slave to Original Master 44](#_Toc346031020)

[4.3.4 xBR or xTP to VIP Failover 44](#_Toc346031021)

[4.3.5 xBRMS to VIP Functionality 45](#_Toc346031022)

[4.3.6 Web UI to VIP Port 8080 Failover 46](#_Toc346031023)

[4.3.7 Web UI to VIP Port 8090 Failover 47](#_Toc346031024)

[4.4 HA Tests – IDMS 48](#_Toc346031025)

[4.4.1 Basic IDMS Failover - status 48](#_Toc346031026)

[4.4.2 Basic IDMS Failover – xBand lrid Retrieval 48](#_Toc346031027)

[Appendix A – Sample xBRC Port 8080 /heartbeat output 50](#_Toc346031028)

[Appendix B - Sample xBRC Port 8080 /readerlocationinfo Output 50](#_Toc346031029)

# Introduction

This document describes the configuration settings of an F5 BigIP virtual appliance for purposes of enabling High Availability for the xConnect project.

## Definitions

For this document, the following acronyms are used:

|  |  |
| --- | --- |
| **Term** | **Definition** |
| VIP | Virtual IP address, aka Flag IP or virtual HA server IP address |
| DIP | Direct IP address, aka underlying server (an actual system) |
| xBR | Long range RFID reader with uni- or omni-directional antennae |
| xTP | Experience TouchPoint, a Disney-themed short range RFID reader or “touch” device |
| DAP | Disney Access Portal, an xTP configured with a biometric reader |
| xBRC | Code package that manages xBRs, xTPs, and DAP devices |
| IDMS | Code and database storing Guest and MagicBand information |
| GxP | Code that handles entitlements |
| HTTP | Hypertext Transfer Protocol |
| Self IP | Management addresses for the BigIP system |
| xBRMS | Code and database which stores operational data |
| xConnect | Code, scripts, APIs, and database schemas which comprise the unifying messaging, management, and reporting software which ties the hardware together into a coherent solution. |

# xConnect Communications

One of the main considerations when considering the proper High Availability (HA) configuration for the xConnect system is how the various components communicate with each other. This chapter describes in general terms the underlying activities.

## xConnect Authentication Subsystem

The following diagrams show how xConnect authentication works. The user interacts with xConnect through a web browser, which in turn talks to an xBRC or XBRMS system, which in turn talks to an IDMS and/or Keystone server.

There are four basic scenarios:

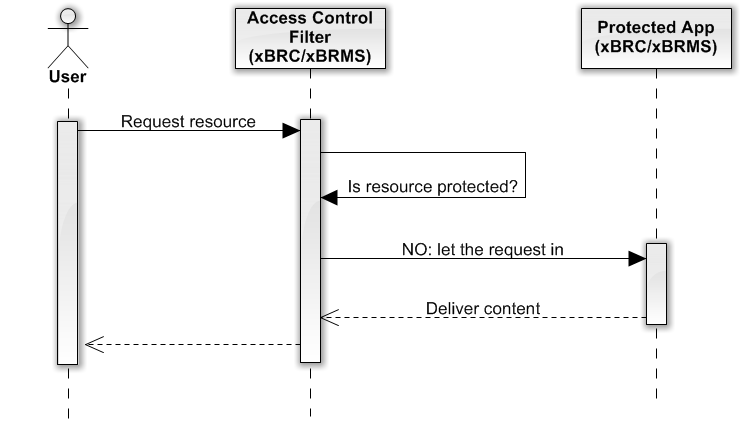


Figure : User requests an unprotected resource

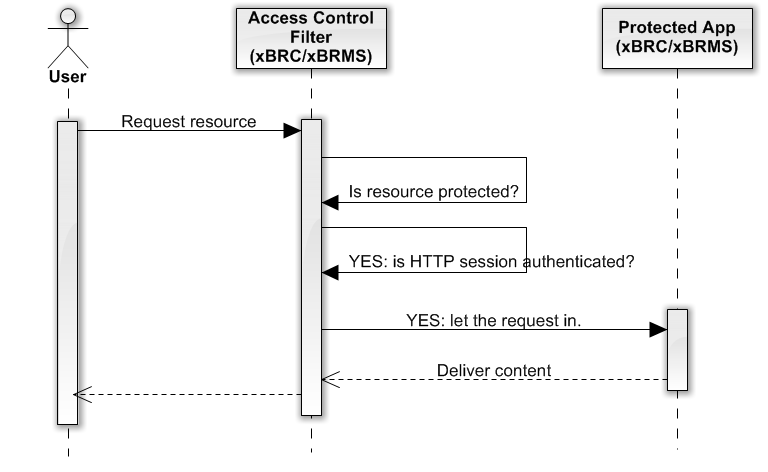


Figure : Request for a protected resource with a previously authenticated application session

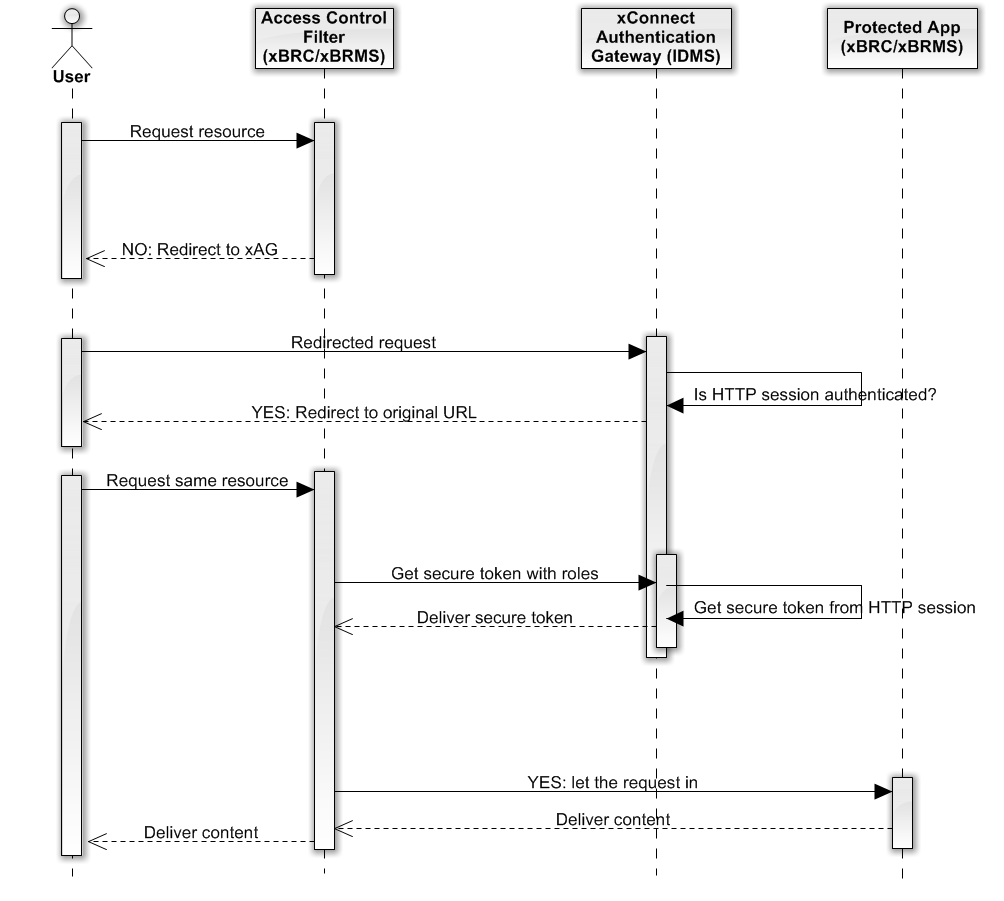


Figure : Request for a protected resource with a previously authenticated xAG session

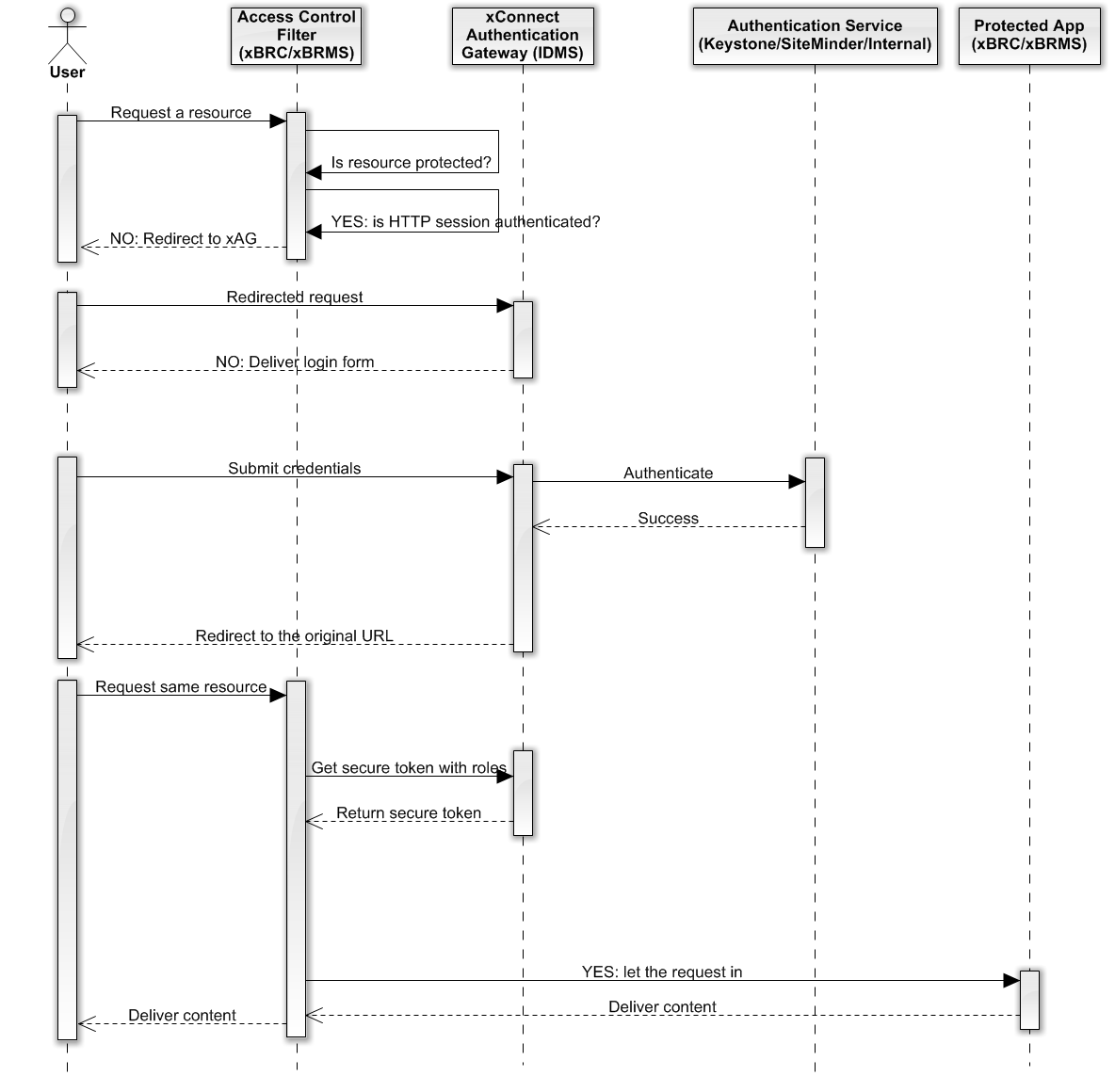


Figure : User request where an xAG HTTP session does not exist

# Configuring F5 BigIP for HA

The BigIP system has a number of configuration settings that need to be defined for it to provide High Availability (HA) capabilities for the xConnect system. This chapter goes through each of the settings.

## Overall HA Configuration

The following diagram shows the configuration of the xBRCs and IDMS systems with respect to internal and external VLANs, and with respect to the nature of the systems to the BigIP system.

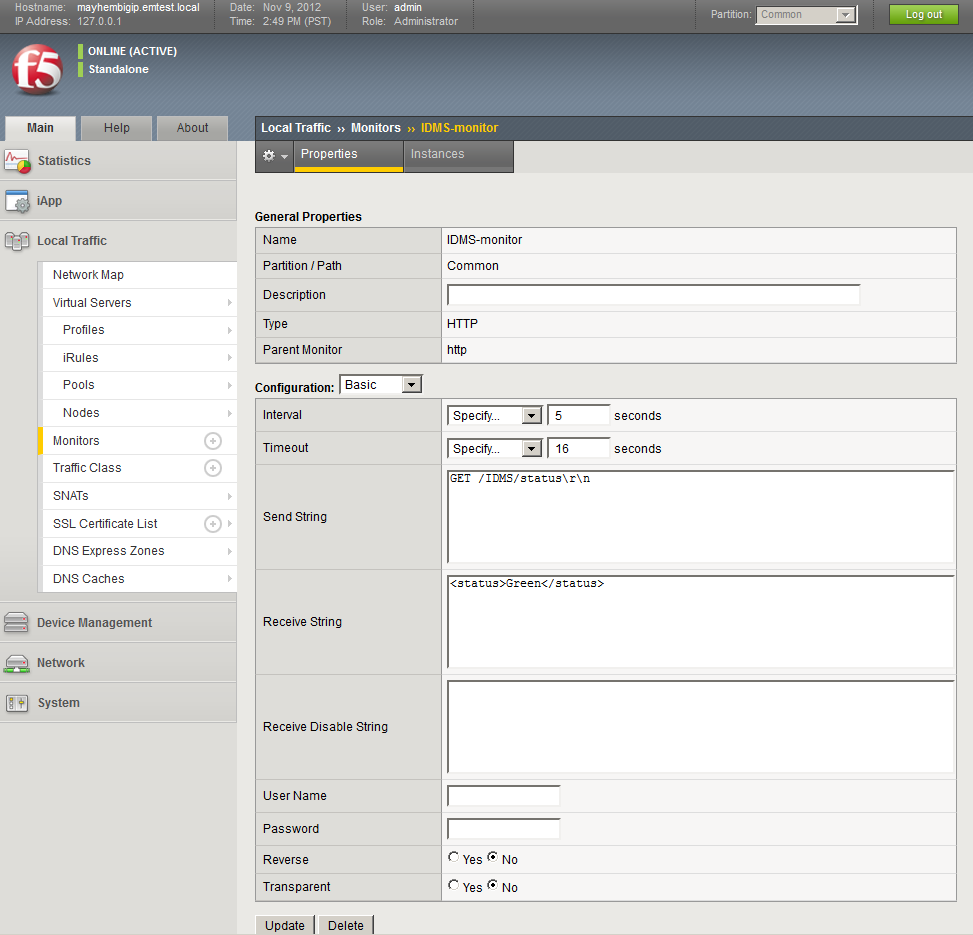


## Setting Up Monitors

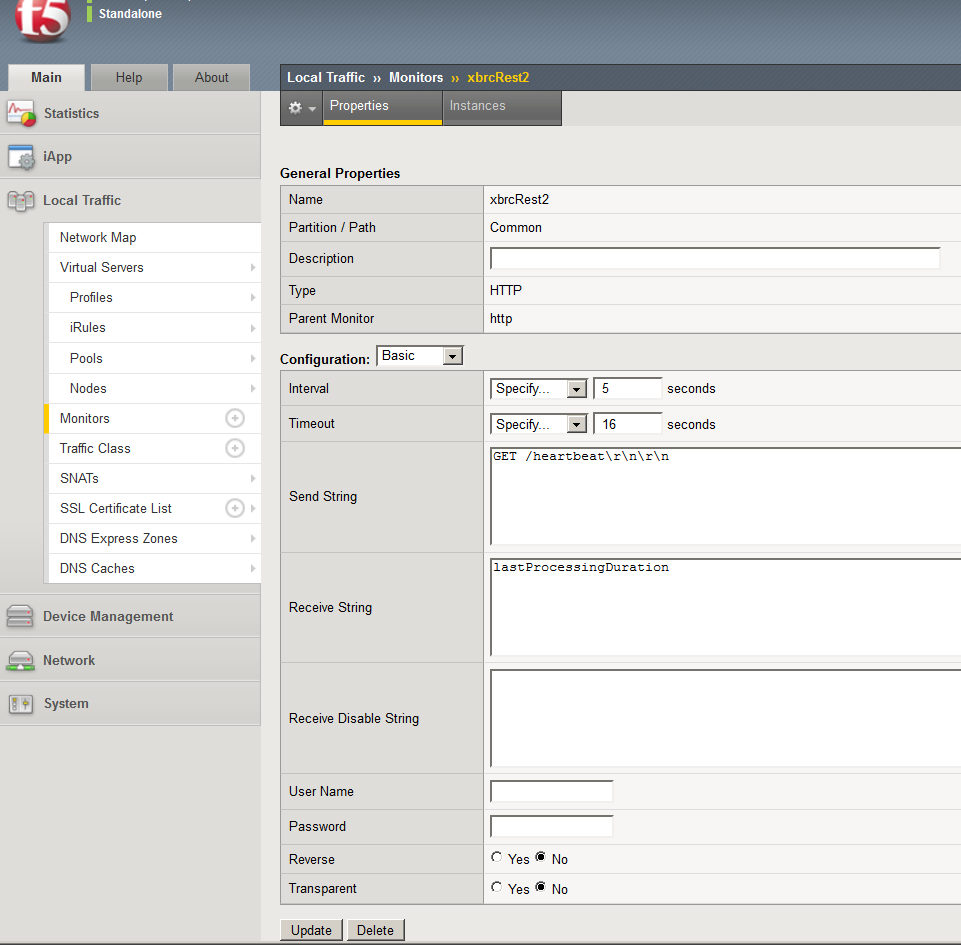
The BigIP uses monitors to determine when to mark a given server as down. If the monitor succeeds, the system is marked as being up. If the monitor fails, the system is marked as being down. A number of standard monitors are provided.

The xConnect system requires three custom monitors to be defined, one for IDMS and two for the xBRCs.

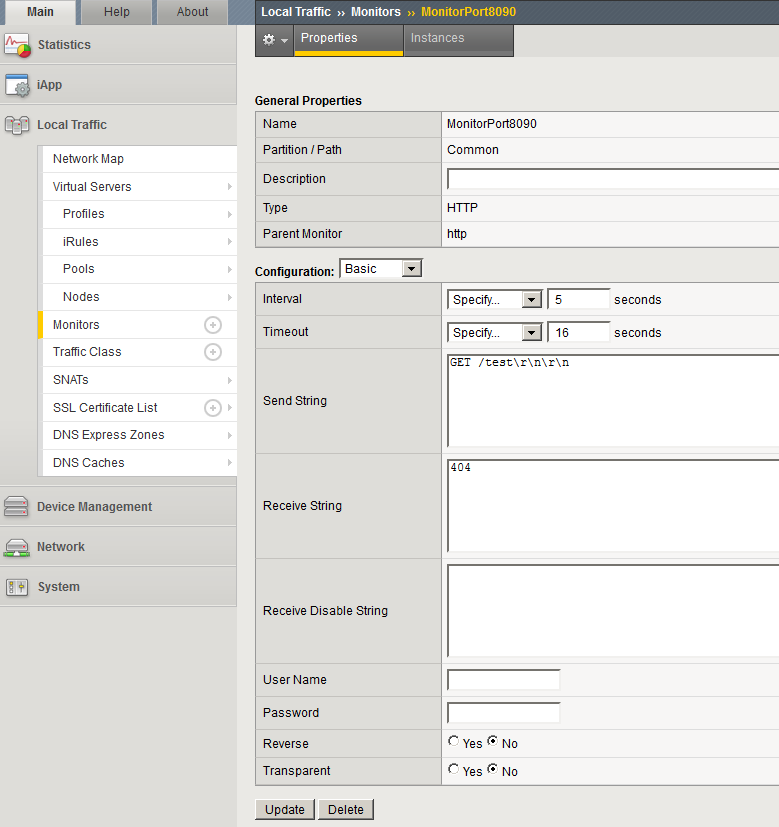
The IDMS monitor contacts the **/IDMS/status** endpoint of the target system, and expects the string “**<status>Green</status>**” to be present in the response:



The xBRC port 8080 monitor contacts the **/heartbeat** endpoint of the target system, and expects the string “**lastProcessingDuration**” to be present in the response:

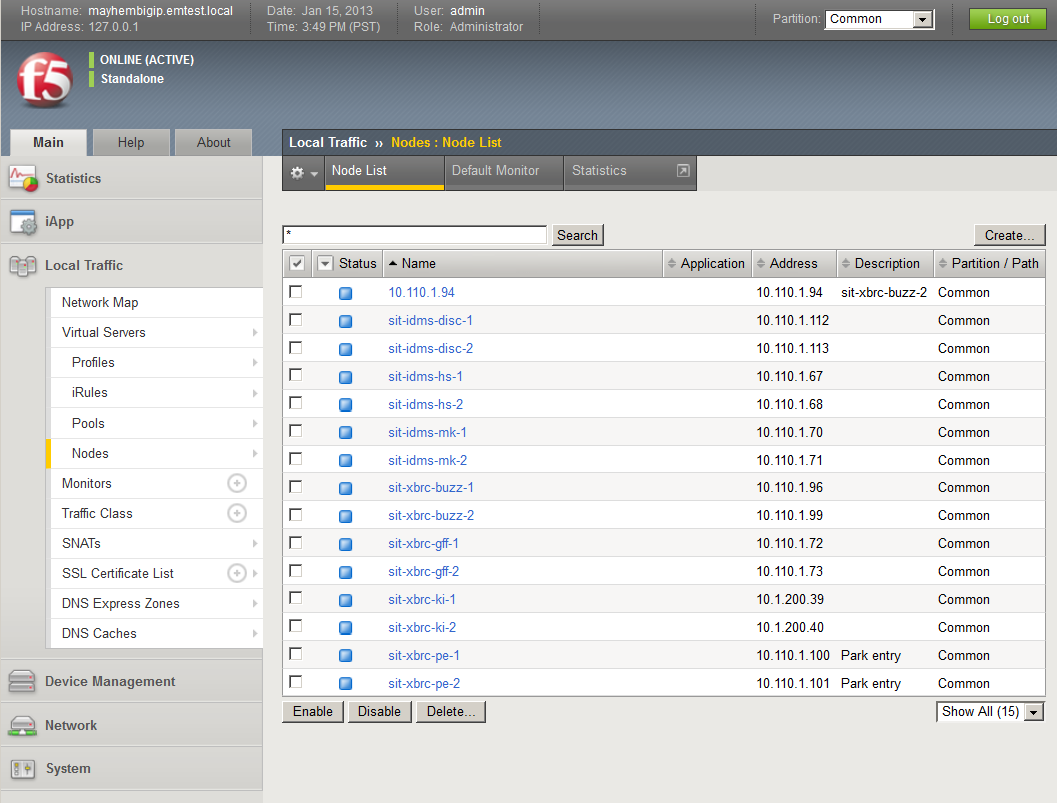


The xBRC port 8090 monitor contacts the **/test** endpoint of the target system, and expects the string “**404**” to be present in the response:



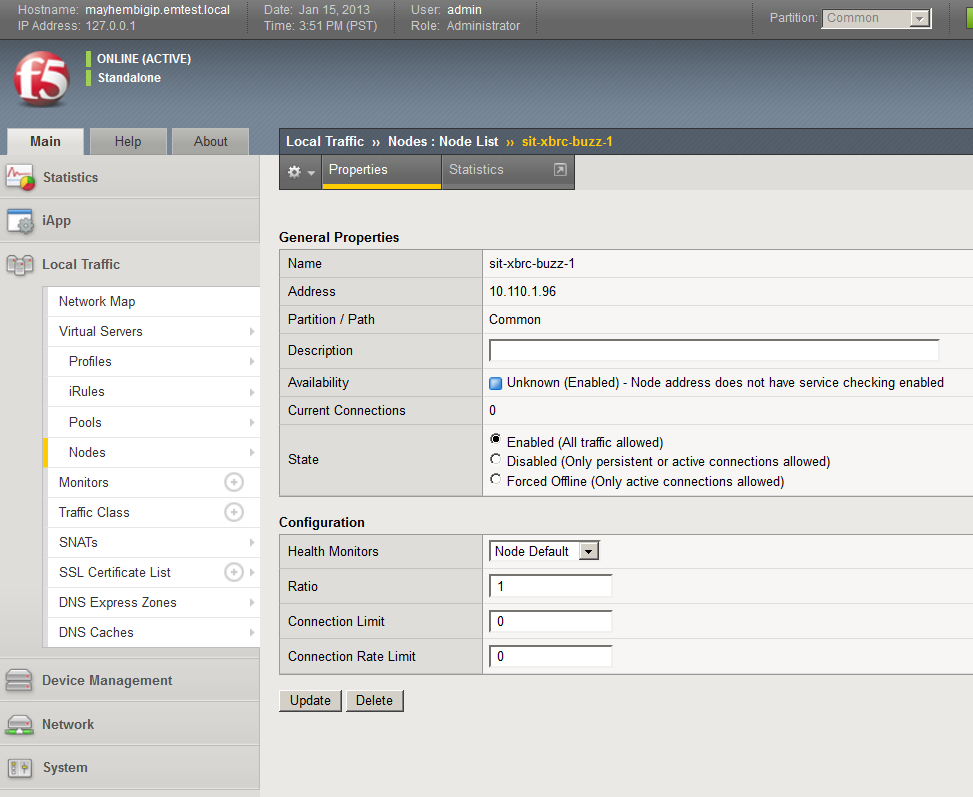
## Setting Up Nodes

A node is an actual server that responds to requests passed through the BigIP.



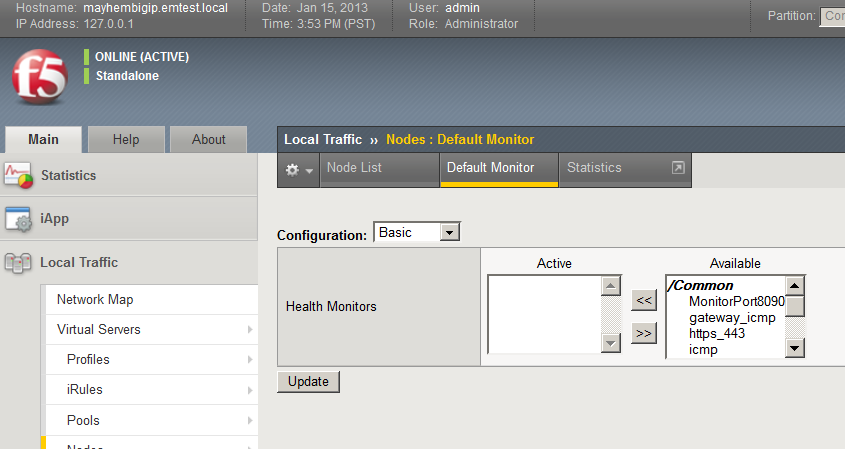
Note in this example that there are two systems for IDMS for each location, i.e., hs (Hollywood Studios), mk (Magic Kingdom), disc (Disney Corporate), and two systems for each xBRC (buzz and gff). Ignore the 10.110.1.94 node.

Here is one of the xBRC nodes. Note that it does not use the standard Health Monitor (gateway\_icmp); hence, the color of the node’s availability is blue. This is separate from the pool monitors which will be used in a separate configuration step.



The other nodes, including the IDMS nodes, use identical configuration settings.

Lastly, the default monitor for the nodes should not be defined.

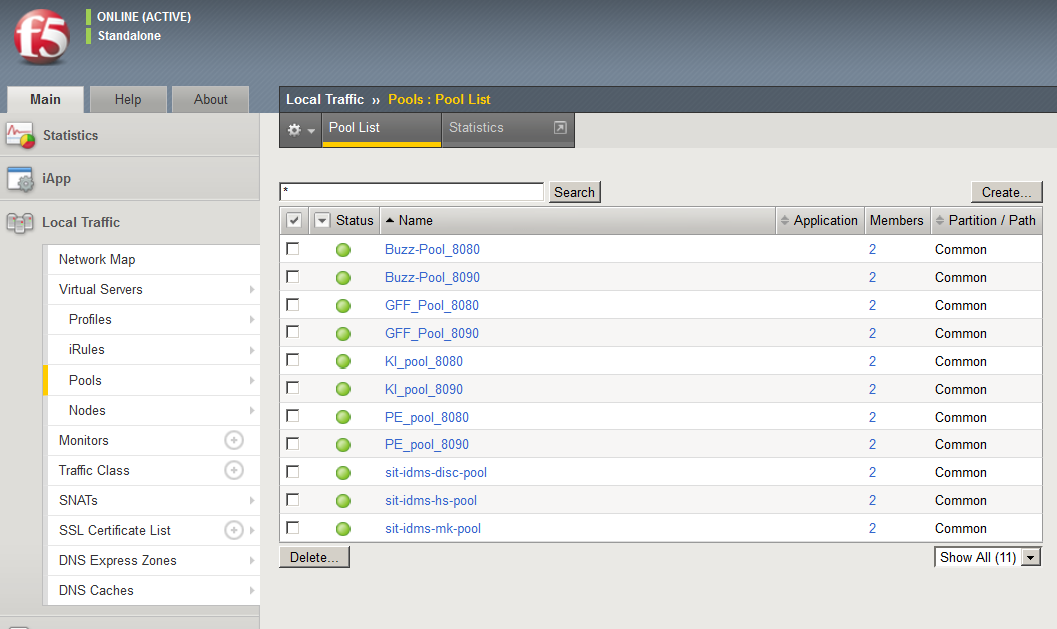


## Setting Up a Pool

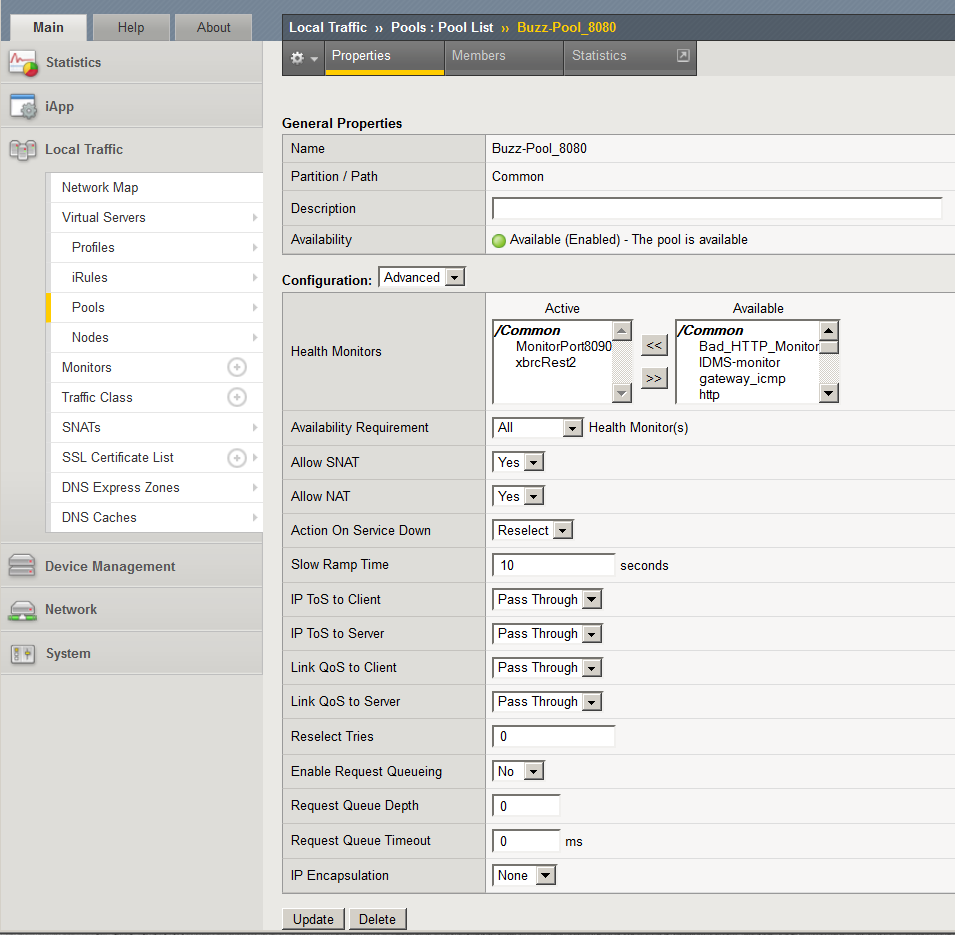
Pools of nodes are used to provide HA capability. When a request is sent to a VIP, it is directed to a node in the pool defined to service that VIP. How a node is chosen depends on the virtual server configuration and node availability.

For xConnect we define two pools for each xBRC, one to service activity to port 8080 and the other to port 8090. With two separate pools we can separately provide monitoring and service to the xBRC application and tcserver. For IDMS systems we use a single pool to service port 8080.

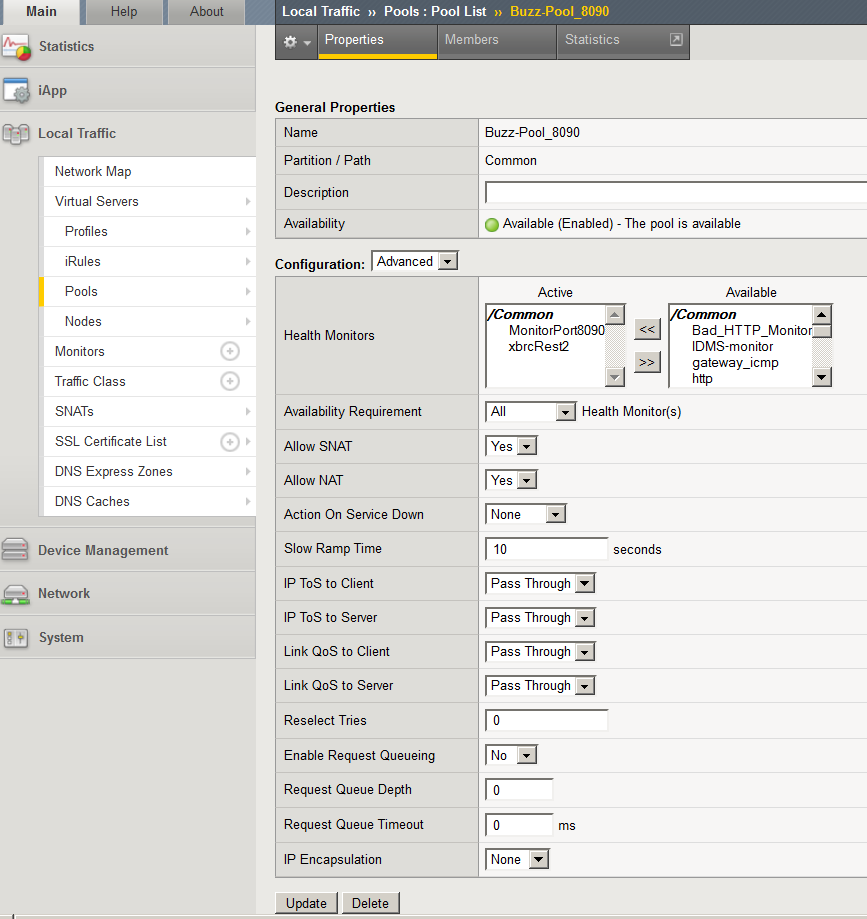
We configure the monitors for xBRCs so that if either service goes away then that particular node is marked down.



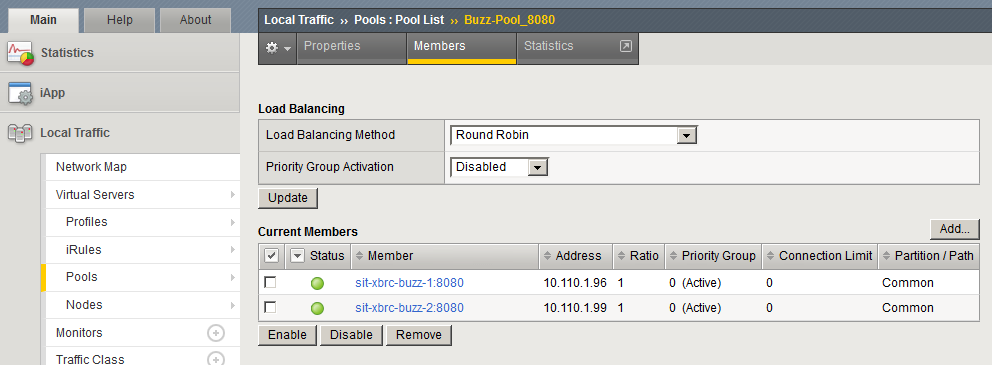
Here are the settings for the xBRC port 8080 pool. We are using the xbrcRest2 and MonitorPort8090 monitors mentioned earlier, and both Allow SNAT and Allow NAT are set to Yes, and Reselect on Service Down.



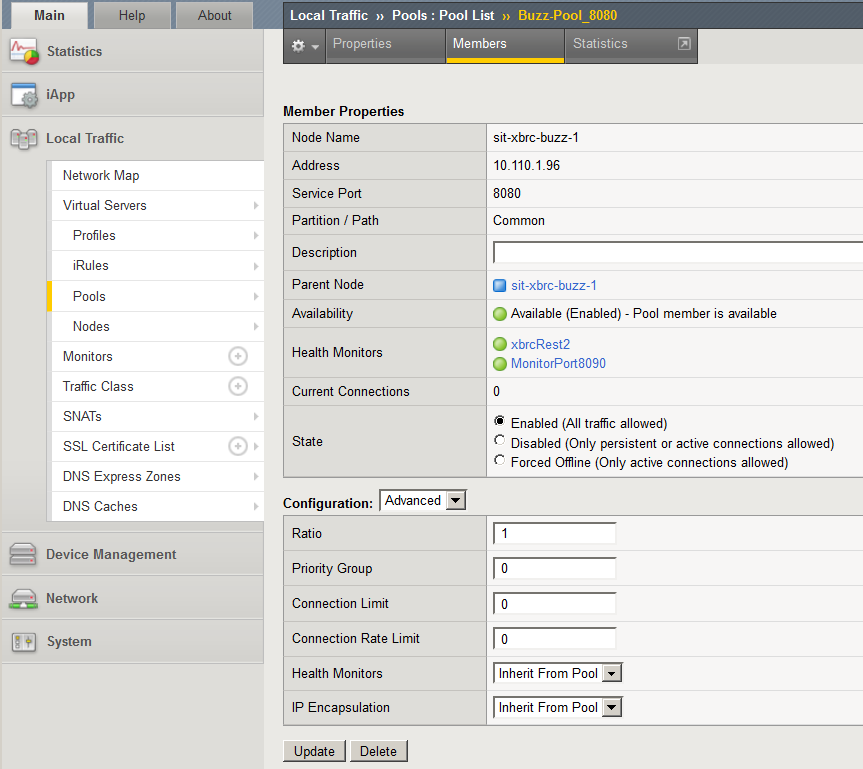
The 8090 pool uses different settings, such as a port of 8090 in place of 8080, but the same Health Monitors. Here are the settings for Buzz-Pool\_8090:



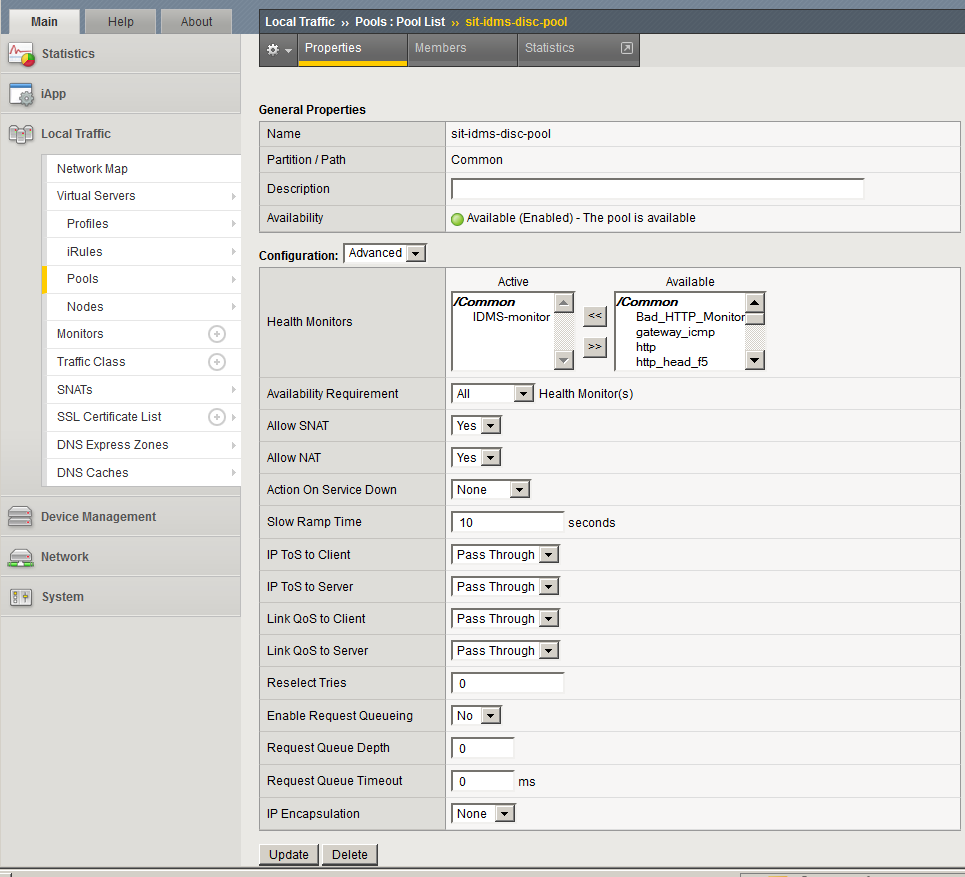
As mentioned earlier, the pool has two members:



The xBRC nodes are defined as follows:

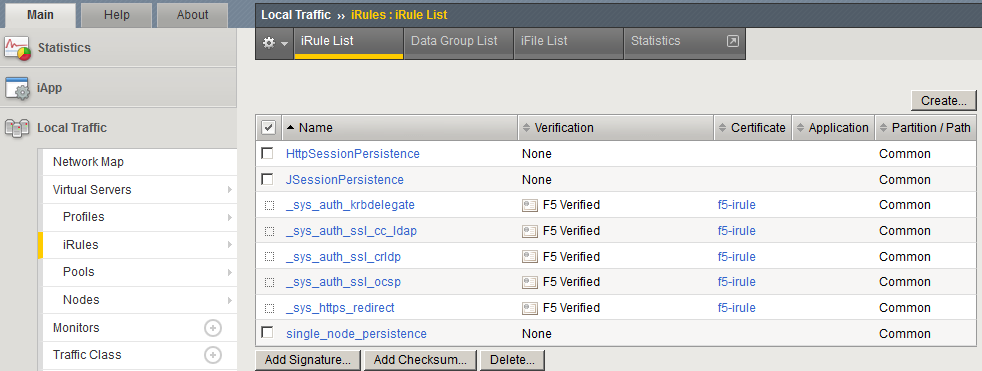


The IDMS pool uses almost identical settings, with the exception being the IDMS-monitor health monitor:



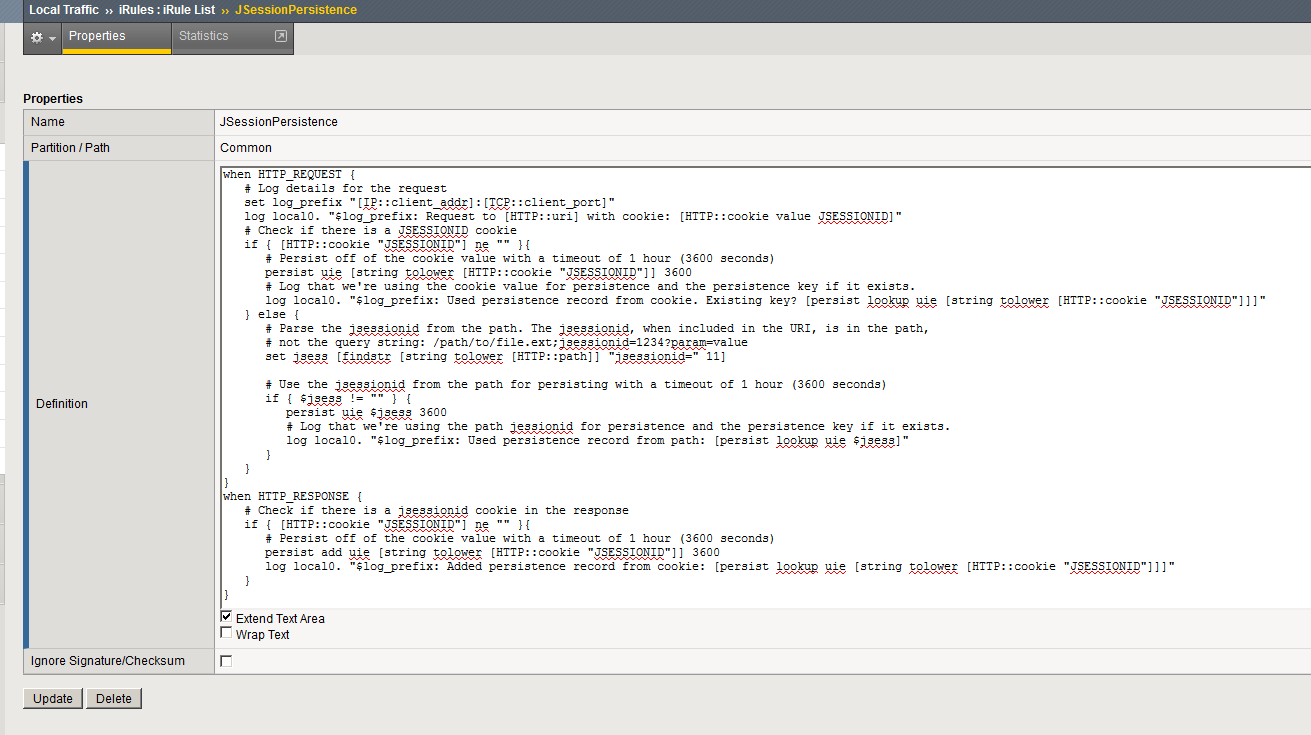
## Defining HA iRules

There are two iRule definitions used in the xConnect configuration of the BigIP. These relate to the persistence of the connection between the browser and/or servers.

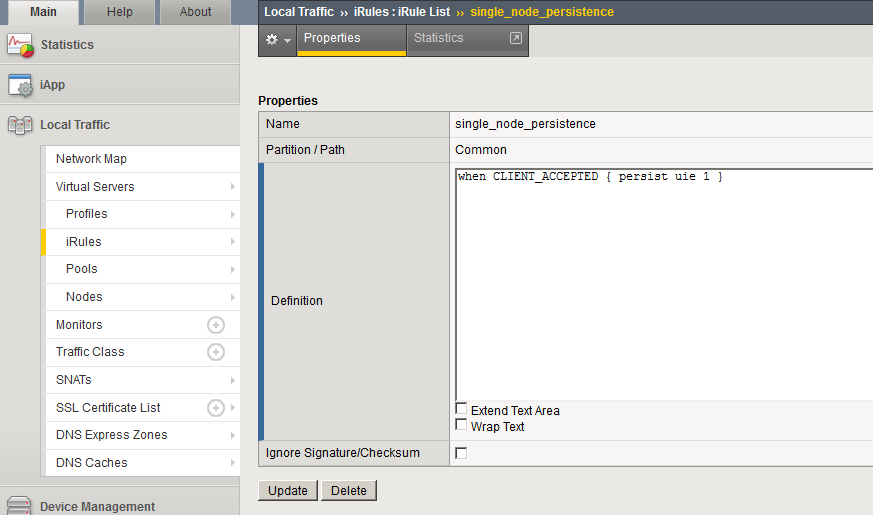


The two iRules we use are the JSessionPersistence and single\_node\_persistence. The JSessionPersistence iRule is for IDMS authentication. The single\_node\_persistence iRule is to handle xBRC communications.

Here is JSessionPersistence definition:



Here is the single\_node\_persistence iRule:



### JSessionPersistence

Here is the JSessionPersistence configuration in machine readable format:

when HTTP\_REQUEST {

# Log details for the request

set log\_prefix "[IP::client\_addr]:[TCP::client\_port]"

log local0. "$log\_prefix: Request to [HTTP::uri] with cookie: [HTTP::cookie value JSESSIONID]"

# Check if there is a JSESSIONID cookie

if { [HTTP::cookie "JSESSIONID"] ne "" }{

# Persist off of the cookie value with a timeout of 1 hour (3600 seconds)

persist uie [string tolower [HTTP::cookie "JSESSIONID"]] 3600

# Log that we're using the cookie value for persistence and the persistence key if it exists.

log local0. "$log\_prefix: Used persistence record from cookie. Existing key? [persist lookup uie [string tolower [HTTP::cookie "JSESSIONID"]]]"

} else {

# Parse the jsessionid from the path. The jsessionid, when included in the URI, is in the path,

# not the query string: /path/to/file.ext;jsessionid=1234?param=value

set jsess [findstr [string tolower [HTTP::path]] "jsessionid=" 11]

# Use the jsessionid from the path for persisting with a timeout of 1 hour (3600 seconds)

if { $jsess != "" } {

persist uie $jsess 3600

# Log that we're using the path jessionid for persistence and the persistence key if it exists.

log local0. "$log\_prefix: Used persistence record from path: [persist lookup uie $jsess]"

}

}

}

when HTTP\_RESPONSE {

# Check if there is a jsessionid cookie in the response

if { [HTTP::cookie "JSESSIONID"] ne "" }{

# Persist off of the cookie value with a timeout of 1 hour (3600 seconds)

persist add uie [string tolower [HTTP::cookie "JSESSIONID"]] 3600

log local0. "$log\_prefix: Added persistence record from cookie: [persist lookup uie [string tolower [HTTP::cookie "JSESSIONID"]]]"

}

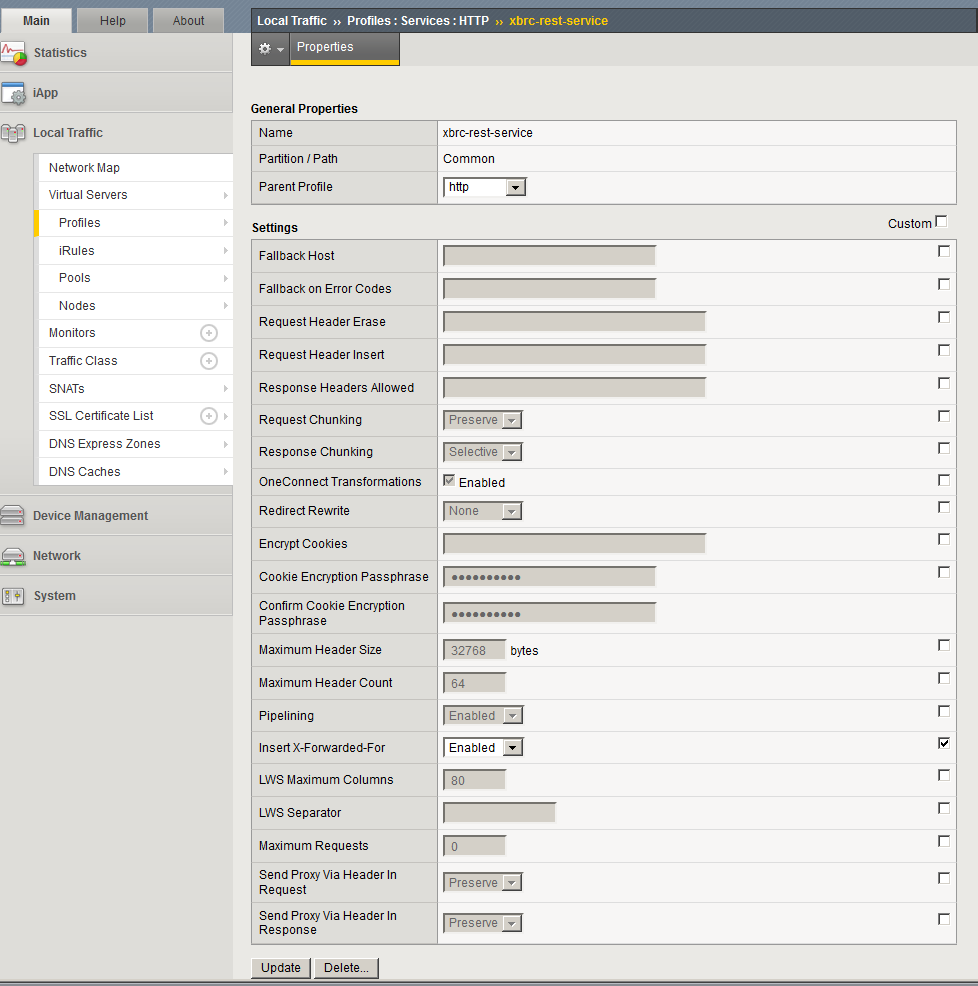
### SingleNodePersistence

Here is the Single node persistence iRule in machine readable format:

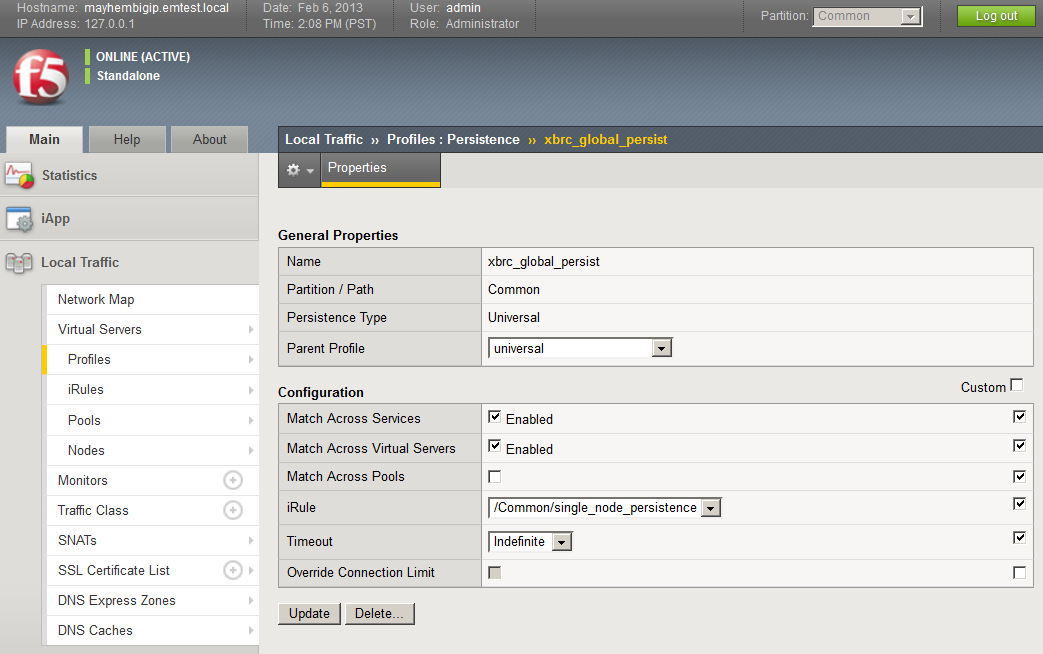
when CLIENT\_ACCEPTED { persist uie 1 }

## Profiles

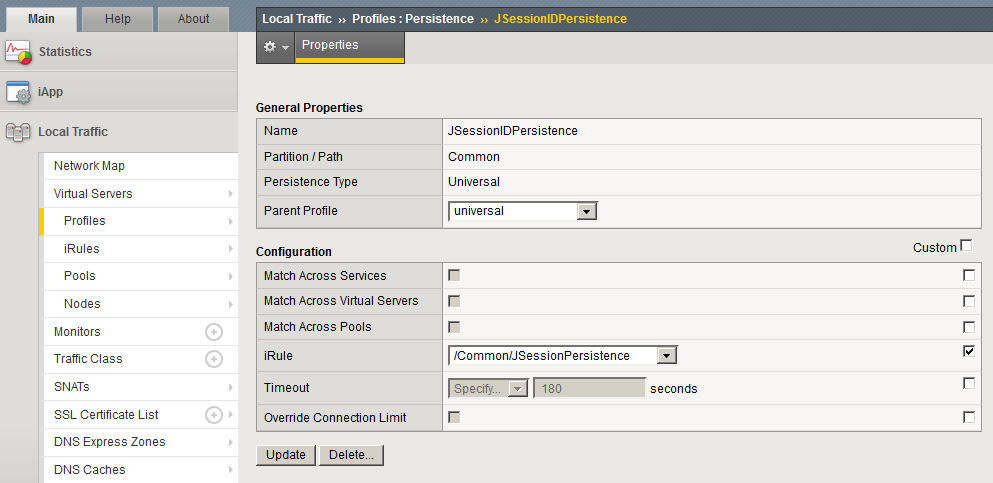
A single profile is defined for HTTP access. It is called xbrc-rest-service and it is used to handle the X-FORWARDED-FOR feature of HTTP:



A second profile is defined to handle the single node persistence. It uses the universal parent profile, and requires matching across services and Virtual Servers. The timeout is set to indefinite:

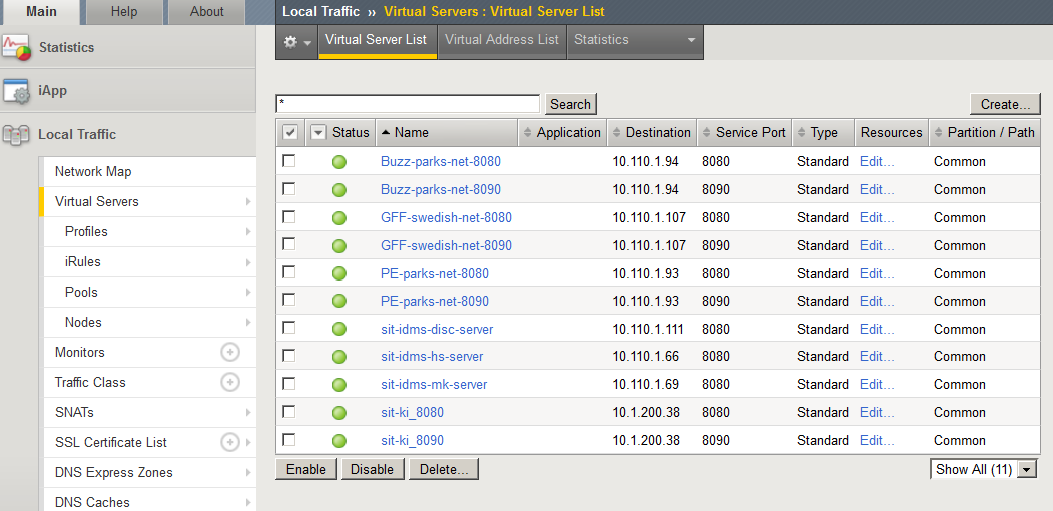


The JSessionIDPersistence Profile is defined as follows with the universal parent and the JSessionPersistence iRule defined previously:



## Setting up a Virtual Server

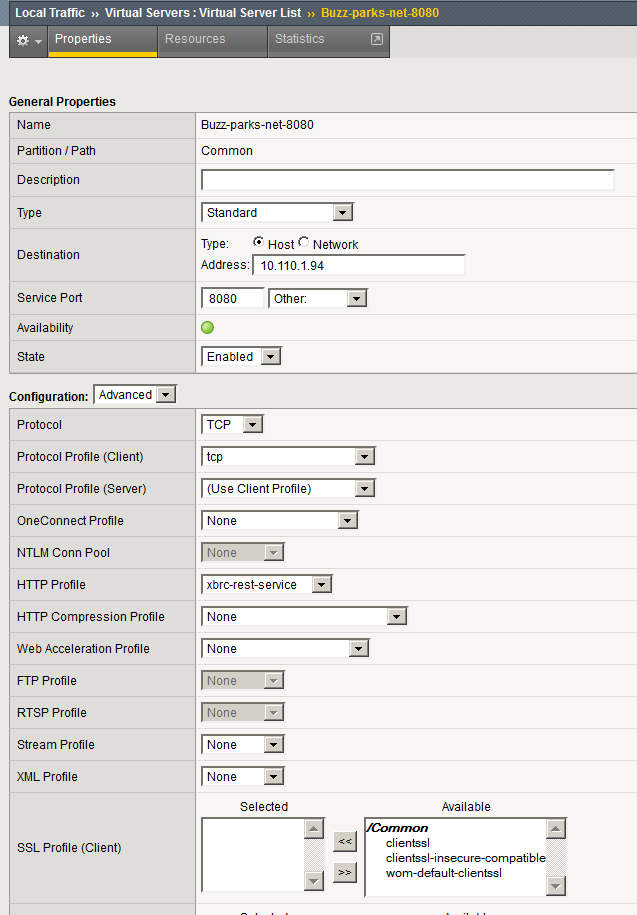
Several virtual servers are defined, one for each service:

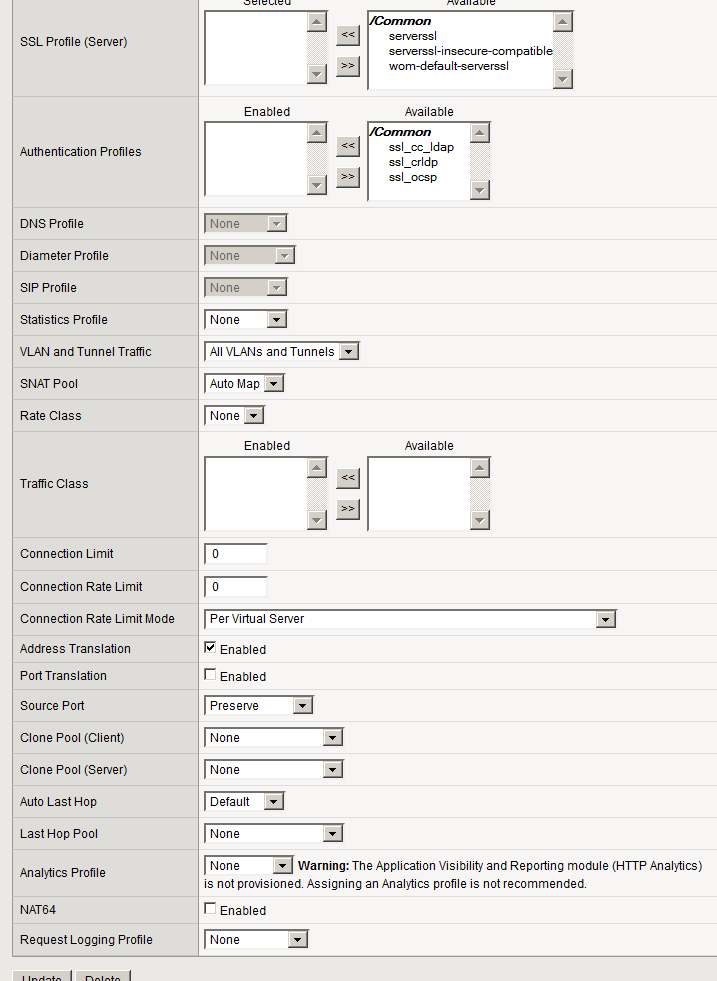


Each of these servers is listed in the following sections.

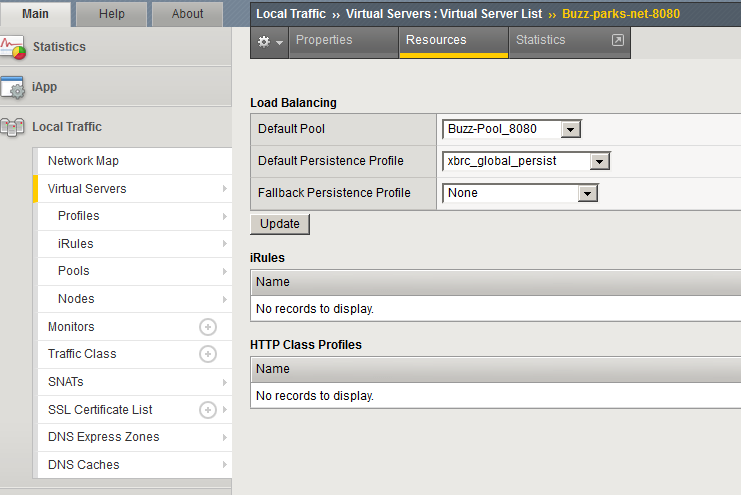
### Buzz-parks-net-8080

The special settings for this virtual server are the HTTP Profile (xbrc-rest-service) and not enabling port translation. Address translation is enabled, however, and SNAT Pool is set to automap.



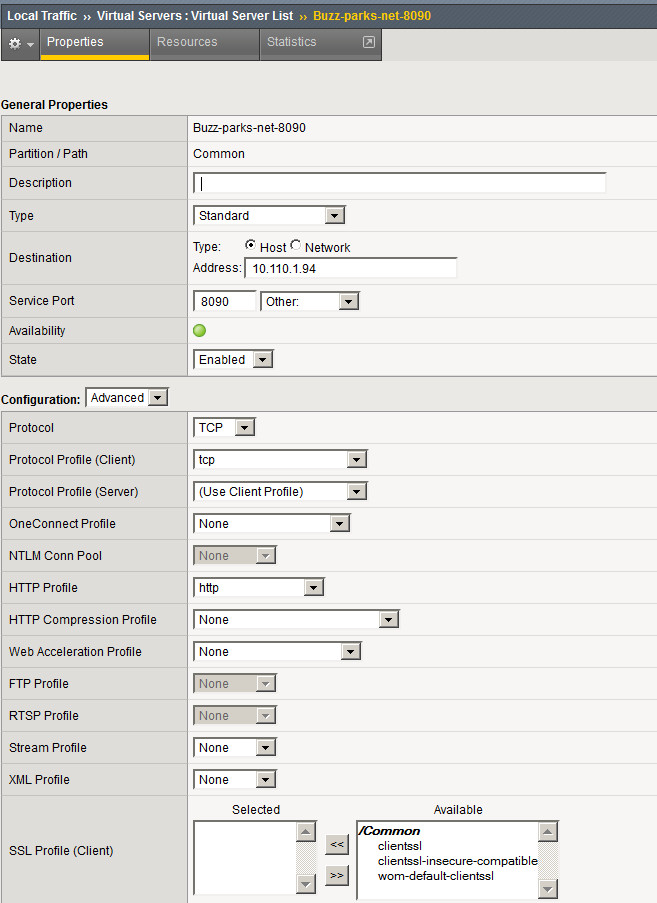


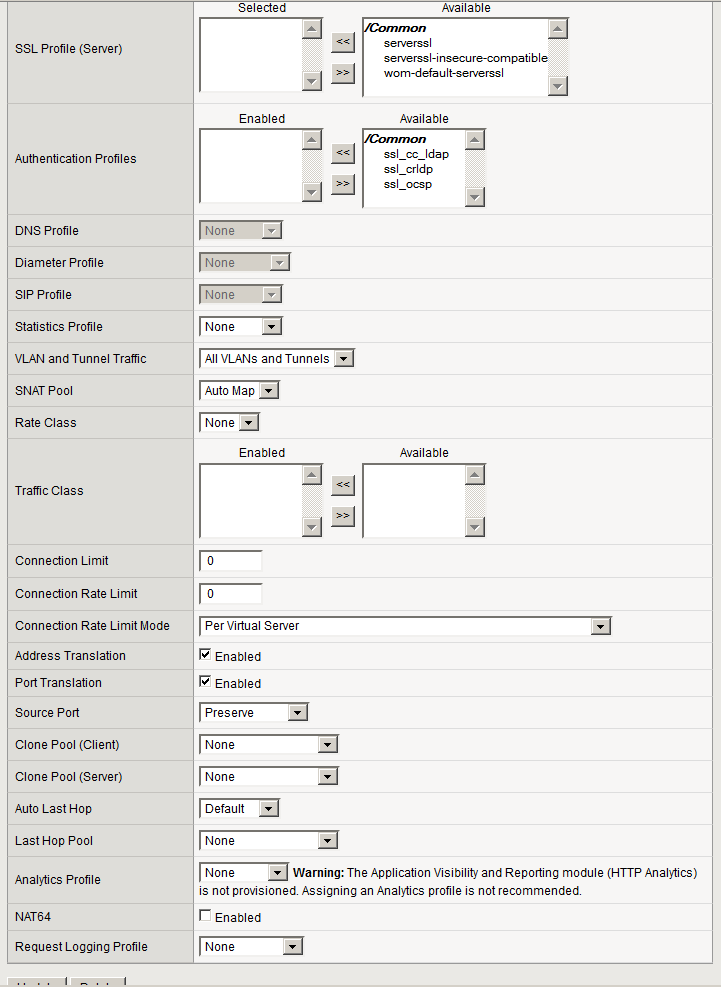
This virtual server uses the following resources:



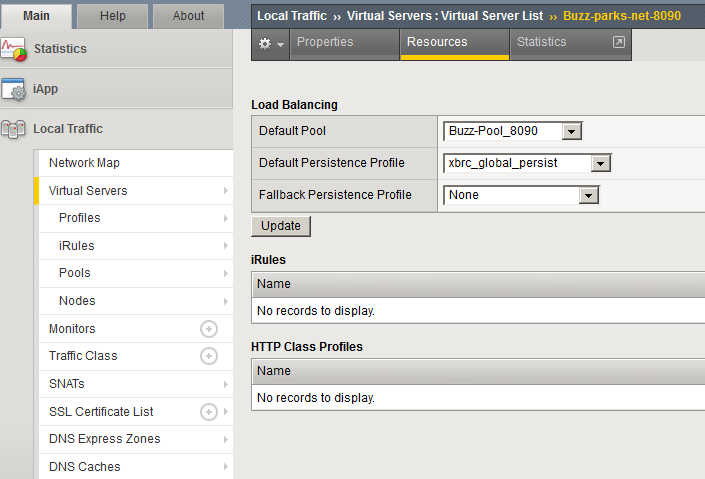
### Buzz-parks-net-8090 Virtual Server

This virtual server has the following configuration. It is almost identical to the settings for the buzz-parks-net-8080 Virtual Server, with the exception of the settings for Port Translation, the Service Port, the http profile and the resource pool.



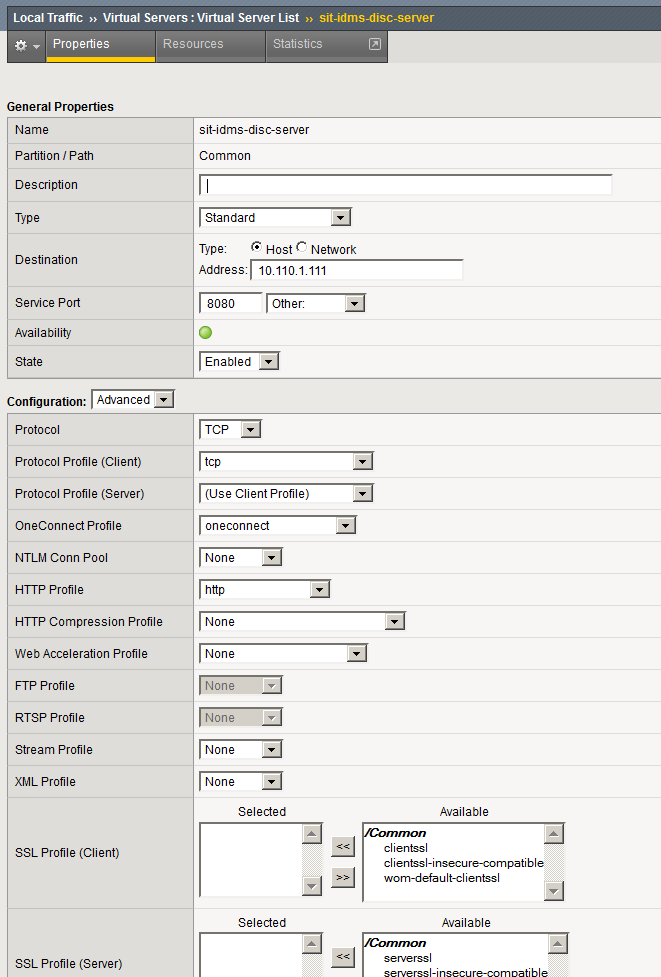


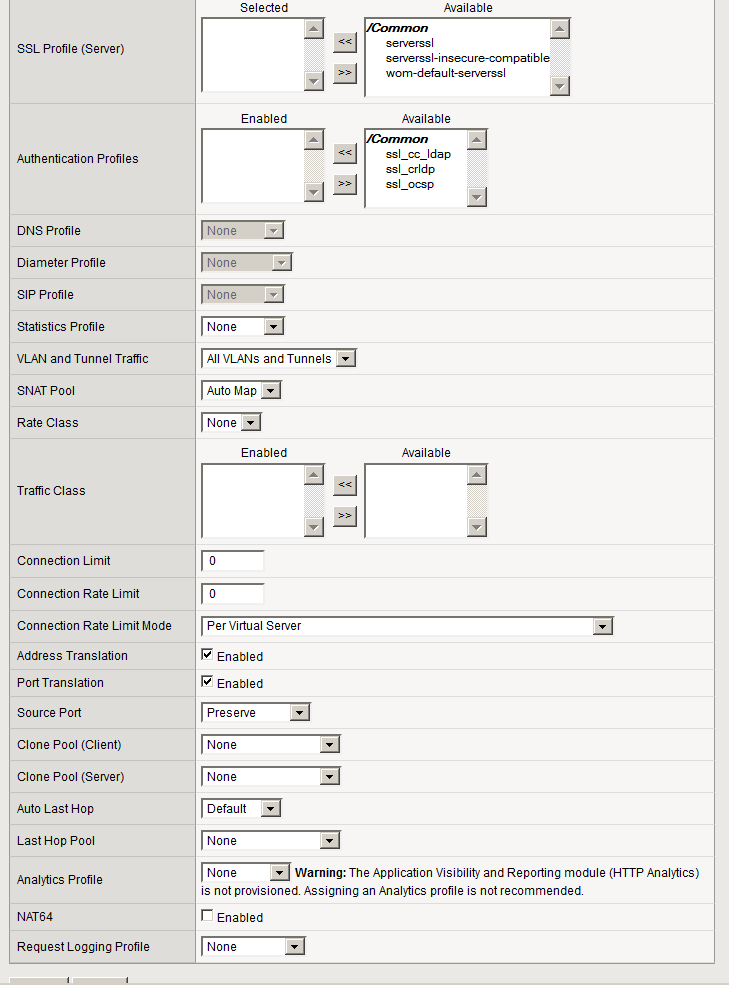
This virtual server uses the following resources:



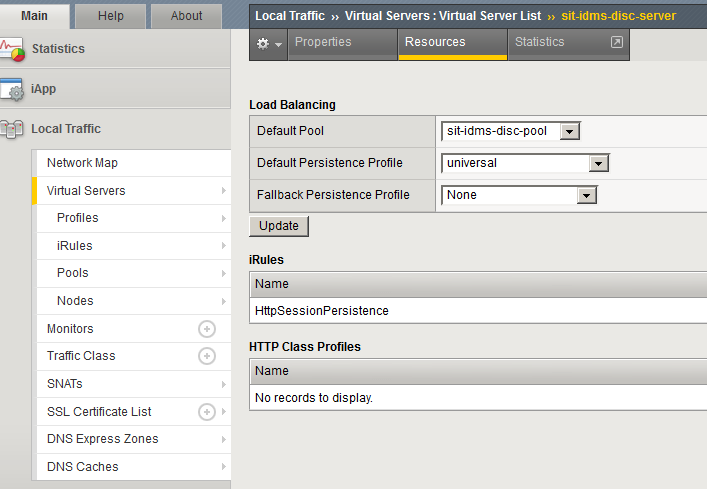
### Sit-idms-disc-server Virtual Server

This virtual server has the following configuration. Note that this server uses the OneConnect Profile and HTTP-session-persistence.





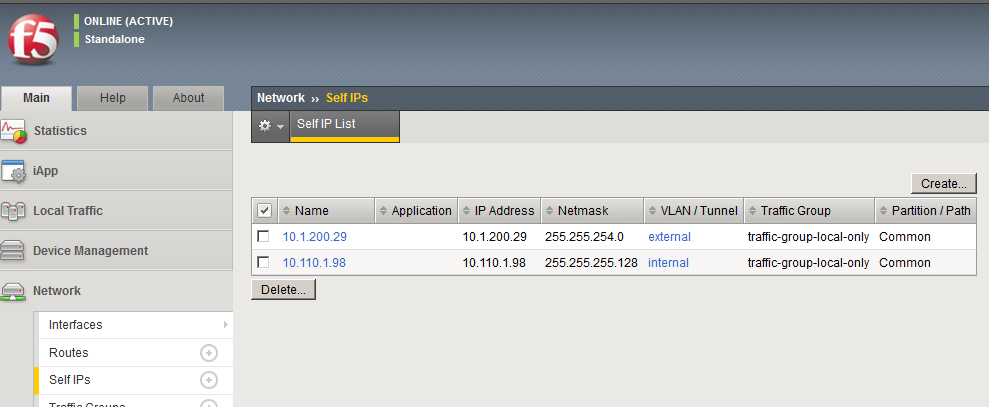
It uses the following resources:



All of the other IDMS virtual servers have the same sort of configuration.

## BigIP Self IPs

At this time two self IP addresses are defined for the BigIP, one on each VLAN.



# Testing F5 BigIP for HA

## Verify Basic Functionality

Verifying that the configuration works prior to testing HA functionality is required as well as straightforward.

Access the DIP endpoints via port 8080

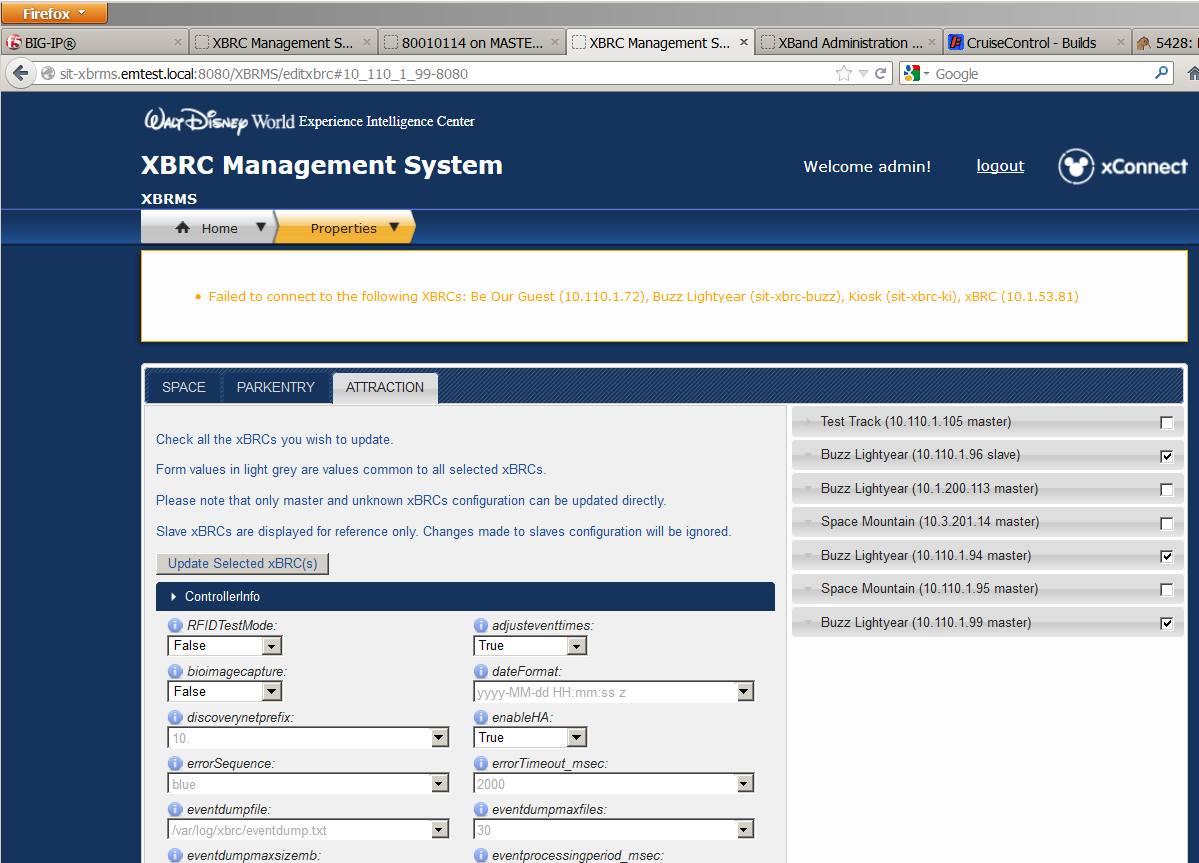
Access the VIP endpoints via port 8080

If XBRCs, then access the DIP endpoints via port 8090

If XBRCs, then log out of the XBRC and access the VIP endpoints via port 8090

In XBRMS, add the systems (DIPs and VIPs) to the System Health page

Verify that the System Health page shows the proper Master and Slave status for each XBRC. If the Master and Slave status are not achieved, verify that the **enableHA** configuration setting is enabled (set to true) in the XBRC configuration page.



## Verify Pool Monitors Are Correct

High Availability functionality requires being able to determine when a member of the HA pool is working properly. When a member is not working properly, its status in the BigIP displays should change to disabled – a red or black color.

The following operations can be run:

* Stop the service (IDMS, XBRC, tcserver, etc.)
* Shutdown the VM
* Disable/enable the network (via ifconfig or vCenter)
* Disable the pool member (via BigIP)

After the timeout period has elapsed the pool member should be marked offline.

Conversely, the monitor should also be able to detect when the HA pool member has returned to operational status. This can be done by doing the opposite of the step(s) performed above. After the timeout period has elapsed the pool member should be marked online.

## HA Tests - XBRCs

Use XBRMS to monitor the status of the master/slave relationship of the xBRCs and the status of the VIP. Otherwise, monitoring port 8080 with the /heartbeat endpoint can provide somewhat faster feedback, with minimal volume of output.

The following test categories can be run:

Master to Slave failover

Slave to original Master (failover the new Master after the original Master returns to service)

xBR or xTP to VIP failover

xBRMS to VIP failover

Web UI to VIP port 8080 failover

Web UI to VIP port 8090 failover

### Master to Slave Failover

This is the most basic HA test; i.e., when the master system becomes unavailable the slave system will switch to become the master and all traffic should be directed to it.

The nature of the xBRC HA software is that if the xBRC receives messages from readers that means the xBRC should be (or become) the master. The BigIP has been configured to determine when an xBRC is working and it will direct reader messages are directed to that xBRC. It will continue sending messages to that xBRC until it determines that the system has failed. At that time it will determine if there are other functional members in the HA pool. If it finds one it will select it to be the recipient of the network traffic.

If there are no active readers, then the time to transition from Unknown to Master or Slave is unpredictable.

Test procedure:

Set up monitoring of the xBRC DIPs and VIPs via XBRMS or via Port 8080 /heartbeat endpoint

Record the settings of the reader locations from either the VIP or the master DIP.

Restart the xBRC services (service xbrc restart)

Wait until the DIPs sort themselves into either Master or Slave mode

Stop the xBRC service on the DIP identifying itself as Master (service xbrc stop)

Wait until the remaining DIPs sort themselves into either Master or Slave mode

Verify that accessing the VIP Port 8080 with the /heartbeat endpoint returns the correct information, e.g., Master

Verify the settings of the reader locations from the new master (and VIP) are the same as prior to the failover

### Slave to Original Master

Continuing with the previous test procedure:

Enable the xBRC service that had been stopped (service xbrc start)

Wait until the DIPs sort themselves into either Master or Slave mode. The newly restarted xBRC should sort itself into Slave mode.

Stop the xBRC service on the DIP identifying itself as Master (service xbrc stop)

Wait until the remaining DIPs sort themselves into either Master or Slave mode

Verify that accessing the VIP Port 8080 with the /status endpoint returns the correct information, e.g., Master

### xBR or xTP to VIP Failover

This test verifies that traffic is directed from a reader to the master xBRC before and after failover.

Test procedure:

Configure the reader to talk to the VIP rather than a DIP.

Configure reader in the xBRC using the VIP address

On each DIP, run “tail –f /var/log/xbrc/eventdump.txt”

Send a series of echo commands to the reader. These should appear in the eventdump.txt file on the Master system.

Stop the xBRC service on the Master system (service xbrc stop)

Send a series of echo commands to the reader. These should appear in the eventdump.txt file on the new Master system.

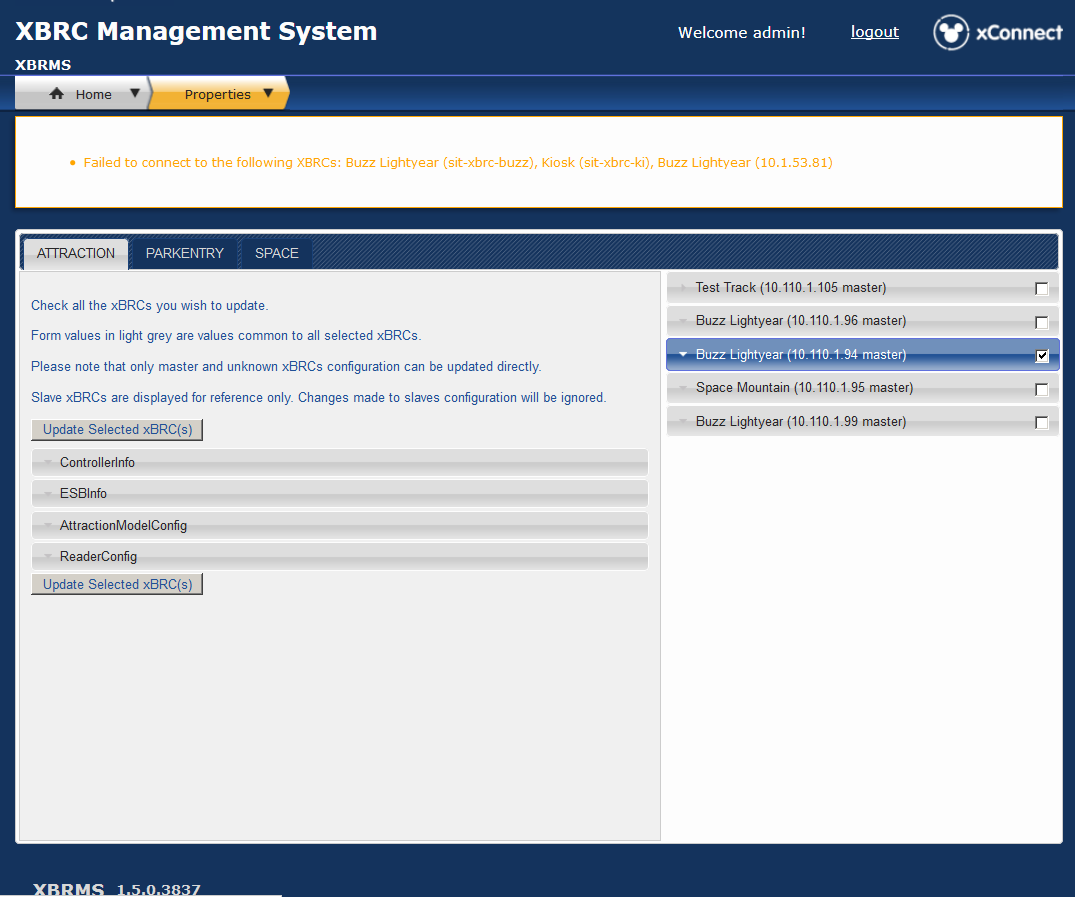
### xBRMS to VIP Functionality

This test verifies that configuration changes made to an xBRC Master get pushed to the Slave.

Test procedure:

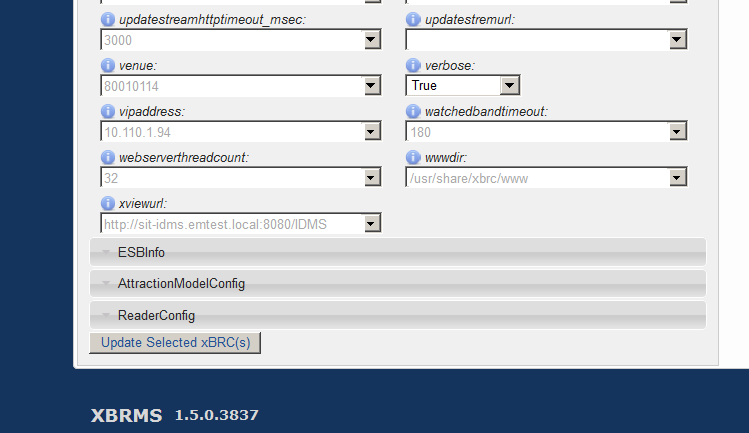
In XBRMS, perform the following steps:

Select the Edit XBRC Properties page

Select the VIP XBRC (check the box)

Select the ControllerInfo field, and expand it

Select vipaddress in ControllerInfo, and set it to the VIP if it is not already set. Otherwise, set it to #.

Click on Update Selected XBRC(s)

Select the DIP XBRCs (check the boxes)

After the UI refreshes, verify that the change in the vipaddress field is retained. Check and uncheck one or more DIP XBRCs and the VIP to make sure the vipaddress field is correct on all related systems.

### Web UI to VIP Port 8080 Failover

This test verifies that access to the web UI is maintained during failover.

Test procedure:

In a web browser, go to http://sit-xbrc-buzz:8080/heartbeat. The results from a sample xBRC are in Appendix A.

Go to http://sit-xbrc-buzz:8080/readerlocationinfo. The results from a sample xBRC are in Appendix B.

Fail over the master system (service xbrc stop).

In a web browser, go to http://sit-xbrc-buzz:8080/heartbeat. The results should be the same as in Appendix A.

In a web browser, go to http://sit-xbrc-buzz:8080/readerlocationinfo. The results should be the same as in Appendix B.

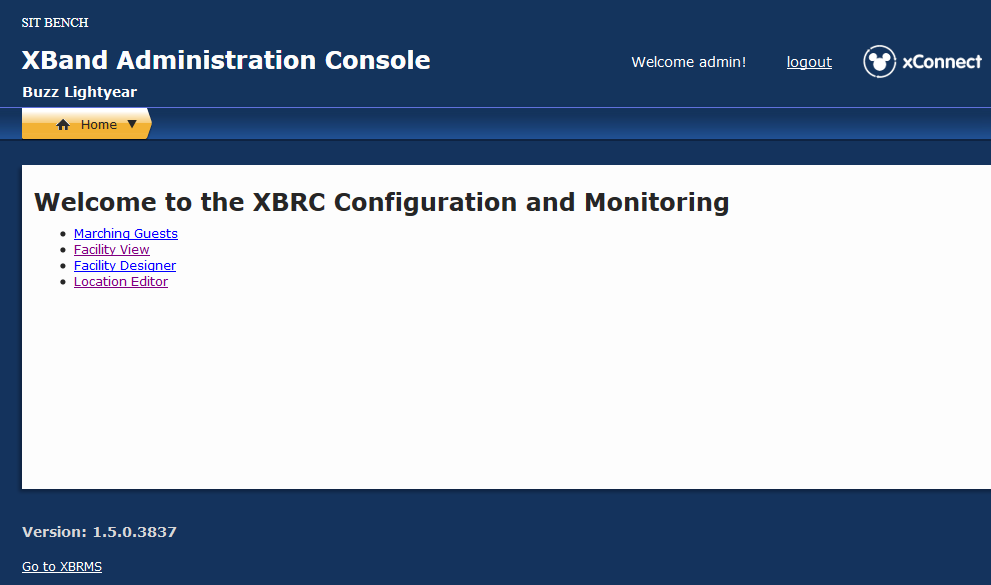
### Web UI to VIP Port 8090 Failover

This test verifies that access to the web UI is maintained during failover.

Test procedure:

In a web browser go to

**http://sit-xbrc-buzz:8090/UI/**

The results should be similar to the following:

Fail over the master system (service xbrc stop).

In a web browser go to

**http://sit-xbrc-buzz:8090/UI/**

The results should be the same as above.

## HA Tests – IDMS

IDMS High Availability testing is fairly complex. It involves verifying database replication from the read/write database located in DISC to the read-only databases located in the HS and MK LDUs. Each of these locations has a pool with at least two servers, with the BigIP system handling failover from one to the other. The testing also involves making sure that IDMS database lookups are handled correctly by the xBRCs and XBRMS, and that changes made to the DISC IDMS database percolate through to the HS and MK IDMS databases within a certain amount of time.

### Basic IDMS Failover - status

Test procedure (to be run on each IDMS VIP):

In a web browser, go to http://sit-idms-mk:8080/status. The results should be as follows:

Hello IDMS

* Host: SIT-IDMS-MK-1
* Version: 1.5.0.3837
* Database Version: 1.5.0.0005
* Status: Green
* Status Message:
* Transaction Count: 79177
* Average Time per Transaction in MS: 21.186341993255617

Fail over the master system (/etc/init.d/bootssptcserver1 stop).

In a web browser, go to http://sit-idms-mk:8080/status. The results should be the same as above, with the exception of the host name and the performance indicators.

### Basic IDMS Failover – xBand lrid Retrieval

Test procedure (to be run on each IDMS VIP):

In a web browser, go to

http://<VIP>:8080/IDMS/xbands/lrid/FFFED90CC0

The results should be as follows:

{"publicId":"35537664","bandType":"Guest","xbandId":"35537664","active":true,"bandFriendlyName":"Cristi Rulison's band","bandId":"35537664","createdBy":null,"createdDate":null,"lrid":"FFFED90CC0","printedName":null,"tapId":"F05E7AAF410400","updatedBy":null,"updatedDate":null,"guests":[{"state":null,"guestId":"35537670","active":true,"createdBy":"simulator","createdDate":1324524636740,"updatedBy":null,"updatedDate":null,"xbands":null,"countryCode":null,"lastName":"Rulison","firstName":"Cristi","address1":null,"address2":null,"birthdate":null,"city":null,"zip":null,"entitlements":null,"guestInfos":null}]}

Fail over the master system (/etc/init.d/bootssptcserver1 stop).

In a web browser, go to

http://<VIP>:8080/IDMS/ xbands/lrid/FFFED90CC0

The results should be the same as above.

Other band IDs that may be used are FFF380A600, FFE8946600, FFF61F5400 and FFEBCE4400.

# Appendix A – Sample xBRC Port 8080 /heartbeat output

<heartbeat>

<lastProcessingDuration>0</lastProcessingDuration>

<mainThreadLoopCount>1702</mainThreadLoopCount>

</heartbeat>

# Appendix B - Sample xBRC Port 8080 /readerlocationinfo Output

<venue time="2012-11-21T21:02:41.152" name="80010190">

<readerlocationinfo><readerlocation>

<section>80010190</section>

<name>Entry</name>

<id>7</id>

<type>7</type>

<typename>xPass Entry</typename>

<x>1.0</x>

<y>1.0</y>

<readers>

<reader>

<name>spce-entry-r</name>

<id>2081133</id>

<type>xfp</type><macaddress>20:08:01:02:03:03</macaddress>

<ipaddress>10.110.1.114</ipaddress>

<port>8090</port>

<gain>1.0</gain>

<threshold>0</threshold>

<lane>3</lane>

<deviceid>2081133</deviceid>

<x>1</x>

<y>1</y>

<timelasthello>1350423210025</timelasthello>

<status>Red</status>

<statusMessage>Reader spce-entry-r not communicating</statusMessage>

<xbioSerialNumber>0</xbioSerialNumber>

<transmitter>false</transmitter>

<transmitPayload>[http://10.110.1.95:8080/</transmitPayload](http://10.110.1.95:8080/%3c/transmitPayload)>

<useSecureId>true</useSecureId>

<hardwareType/>

<signalStrengthTransitThreshold>-127</signalStrengthTransitThreshold>

<enabled>true</enabled>

<bioDeviceType>0</bioDeviceType>

</reader>

<reader>

<name>spce-entry-l</name>

<id>2081144</id>

<type>xfp</type>

<macaddress>20:08:01:02:04:04</macaddress>

<ipaddress>10.110.1.114</ipaddress>

<port>8090</port>

<gain>1.0</gain>

<threshold>0</threshold>

<lane>4</lane>

<deviceid>2081144</deviceid>

<x>1</x>

<y>1</y>

<timelasthello>1350423210026</timelasthello>

<status>Red</status>

<statusMessage>Reader spce-entry-l not communicating</statusMessage>

<xbioSerialNumber>0</xbioSerialNumber>

<transmitter>false</transmitter>

<transmitPayload>[http://10.110.1.95:8080/</transmitPayload](http://10.110.1.95:8080/%3c/transmitPayload)>

<useSecureId>true</useSecureId>

<hardwareType/>

<signalStrengthTransitThreshold>-127</signalStrengthTransitThreshold>

<enabled>true</enabled>

<bioDeviceType>0</bioDeviceType>

</reader>

<reader>

<name>spce-entry-hwl</name>

<id>2154145</id>

<type>xfp</type>

<macaddress>02:09:90:01:09:D8</macaddress>

<ipaddress>10.110.1.11</ipaddress>

<port>8080</port>

<gain>1.0</gain>

<threshold>-90</threshold>

<lane>0</lane>

<deviceid>10110111</deviceid>

<x>0</x>

<y>0</y>

<timelasthello>1353531752350</timelasthello>

<status>Green</status>

<statusMessage/>

<minXbrcVersion>0.0.0.0</minXbrcVersion>

<version>1.2.2-2607</version>

<xbioSerialNumber>0</xbioSerialNumber>

<transmitter>false</transmitter>

<useSecureId>true</useSecureId>

<hardwareType>xTP1</hardwareType>

<signalStrengthTransitThreshold>-127</signalStrengthTransitThreshold>

<enabled>true</enabled>

<bioDeviceType>0</bioDeviceType>

</reader></readers>

<useSecureId>true</useSecureId>

<successTimeout>2500</successTimeout>

<failureTimeout>0</failureTimeout>

<errorTimeout>2000</errorTimeout>

</readerlocation>

<readerlocation>

<section>80010190</section>

<name>Merge</name>

<id>6</id>

<type>6</type>

<typename>Merge</typename>

<x>1.0</x>

<y>1.0</y>

<readers>

<reader>

<name>spce-merge-hwl</name>

<id>29</id>

<type>xfp</type>

<macaddress>02:19:B0:00:09:D8</macaddress>

<ipaddress>10.110.1.40</ipaddress>

<port>8080</port>

<gain>1.0</gain>

<threshold>-90</threshold>

<lane>0</lane>

<deviceid>10110140</deviceid>

<x>0</x>

<y>0</y>

<timelasthello>1353531748314</timelasthello>

<status>Green</status>

<statusMessage/>

<minXbrcVersion>0.0.0.0</minXbrcVersion>

<version>1.2.2-2607</version>

<xbioSerialNumber>0</xbioSerialNumber>

<transmitter>false</transmitter>

<useSecureId>true</useSecureId>

<hardwareType>xTP1</hardwareType>

<signalStrengthTransitThreshold>-127</signalStrengthTransitThreshold>

<enabled>true</enabled>

<bioDeviceType>0</bioDeviceType>

</reader>

<reader>

<name>spce-merge-r</name>

<id>2084133</id>

<type>xfp</type>

<macaddress>20:08:04:02:03:03</macaddress>

<ipaddress>10.110.1.114</ipaddress>

<port>8090</port>

<gain>1.0</gain>

<threshold>0</threshold>

<lane>3</lane>

<deviceid>2084133</deviceid>

<x>1</x>

<y>1</y>

<timelasthello>1350423209993</timelasthello>

<status>Red</status>

<statusMessage>Reader spce-merge-r not communicating</statusMessage>

<xbioSerialNumber>0</xbioSerialNumber>

<transmitter>false</transmitter>

<transmitPayload>[http://10.110.1.95:8080/</transmitPayload](http://10.110.1.95:8080/%3c/transmitPayload)>

<useSecureId>true</useSecureId>

<hardwareType/>

<signalStrengthTransitThreshold>-127</signalStrengthTransitThreshold>

<enabled>true</enabled>

<bioDeviceType>0</bioDeviceType>

</reader>

<reader>

<name>spce-merge-l</name>

<id>2084144</id>

<type>xfp</type>

<macaddress>20:08:04:02:04:04</macaddress>

<ipaddress>10.110.1.114</ipaddress>

<port>8090</port>

<gain>1.0</gain>

<threshold>0</threshold>

<lane>4</lane>

<deviceid>2084144</deviceid>

<x>1</x>

<y>1</y>

<timelasthello>1350423210022</timelasthello>

<status>Red</status>

<statusMessage>Reader spce-merge-l not communicating</statusMessage>

<xbioSerialNumber>0</xbioSerialNumber

><transmitter>false</transmitter>

<transmitPayload>[http://10.110.1.95:8080/</transmitPayload](http://10.110.1.95:8080/%3c/transmitPayload)>

<useSecureId>true</useSecureId>

<hardwareType/>

<signalStrengthTransitThreshold>-127</signalStrengthTransitThreshold>

<enabled>true</enabled>

<bioDeviceType>0</bioDeviceType>

</reader>

</readerlocation>

</readerlocationinfo>

</venue>