xConnect High Availability Troubleshooting Guide

Principles of operation and Troubleshooting guide

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Rev | Date | Author | Description |
| 1.0 | 12/5/2012 | Manny Vellon | Intial Draft |
| 1.6 | 4/18/2013 | Michael Lampi | Updated to xConnect 1.6 |
|  |  |  |  |

**Document Approvers & Sign-Off**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Approver** | **Role** | **Document Accept/Reject** |
| 12/5/2012 | Manny Vellon | Release Manager | Accept |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 4](#_Toc354062955)

[2 HIgh Availability design 4](#_Toc354062956)

[2.1 xBRC HA Architecture 4](#_Toc354062957)

[2.2 Sequence Diagrams 5](#_Toc354062958)

[2.3 XBRMS HA Architecture 7](#_Toc354062959)

[2.4 IDMS HA Architecture 7](#_Toc354062960)

[3 Proper xBRC Configuration for HA 7](#_Toc354062961)

[3.1 /etc/nge/config/environment.properties 7](#_Toc354062962)

[3.2 xBRC Configuration Settings 7](#_Toc354062963)

[4 Troubleshooting xBRC HA problems 8](#_Toc354062964)

[5 Troubleshooting xBRMS HA problems 11](#_Toc354062965)

[6 Troubleshooting IDMS HA problems 12](#_Toc354062966)

# Introduction

Version 1.4 of the xConnect software introduced support for high availability (HA) for xBRCs. Version 1.5 added further enhancements and fixes to the functionality. Version 1.6 added HA enhancements to xBRMS. The testing of the HA code and its rollout in FPT3 have demonstrated the importance of understanding how HA works and the importance of correctly configuring the xConnect sytem to assure that it works properly. This document provides a high-level description of the xBRC HA design and points out the network components and property settings that must be properly configured in order for the design to work. It also provides a set of frequently observed errors and how to diagnose and correct them.

# HIgh Availability design

## xBRC HA Architecture

For much of NGE, high availability is achieved by running multiple instances of a service, typically in different LDU racks, then having an F5 BigIP appliance route traffic to these instances using load balancing techniques. Clients of these services access the service using its *virtual IP* (VIP) address. The BigIP appliance then routes traffic to a specific instance using its *direct IP* (DIP) address.

Load balancing does not work properly for xBRCs. An xBRC instance must receive **all** xBR and xTP events for a band in order to properly singulate guest locations and in order to properly perform its modeling operations; for example, noting that a guest has changed from *HASENTERED* state to *MERGED*. If some reader events were sent to one xBRC instance and other events were sent to a different one, the xBRC would be unable to perform singulation or modeling properly. Load balancing only works when services are relatively stateless, or stateful at the HTTP “session” level. xBRCs are stateful across multiple HTTP sessions - those established with different readers.

The xBRC HA design uses a *master-slave* approach. All reader traffic is routed to a particular xBRC instance. If the instance fails then all traffic is sent to the slave, which then becomes the new master. Figure 1 illustrates the conceptual network architecture for the solution.

Readers communicate with the assigned xBRC by using its VIP. The BigIP appliance routes all traffic to the currently identified xBRC master instance using its DIP. If the master fails, as determined by a BigIP *monitor*, traffic is then routed to the xBRC slave instance.

Recalling that xBRCs are inherently stateful, the design also incorporates state synchronization between the master and slave by employing messages sent across the Sonic JMS bus. The slave also synchronizes certain configuration information by communicating directly with the master using HTTP-based web services.

  
Figure 1 - HA Network Architecture

## Sequence Diagrams

Although the conceptual HA design of the xBRC is simple, implementation of the design requires some careful choreography. The following figure describes how the master and slave instances start up, how information flows between them and how failover occurs.

When an xBRC instance starts up, i.e., the service is started on its VM, it starts with its HA state set to *unknown*. At that point, the xBRC doesn’t know what role it is playing. It is ready to receive events from readers and synchronization messages from the JMS.

As soon as a reader“hello”or data event is routed by the BigIP appliance from a reader to the master xBRC, that xBRC changes its state to *master*. It then sends a *DISCOVERY* message on the JMS bus.

The slave xBRC receives the DISCOVERY message and notes that another xBRC is asserting master status. The slave xBRC changes from *unknown* to *slave* HA status and synchronizes its configuration information with the master xBRC. It also retrieves the guest state from the master, i.e., the initial Guest Status Table (GST).

  
Figure 2 - Startup, Synchronization and Failover

As more events as are received by the master xBRC, it sends messages on the JMS informing the slave (and other downstream applications) of its activity. These messages generally indicate changes to guest state. As such, when the slave receives these messages, it edits its own copy of its GST in order to remain synchronized with the master.

If and when a master fails, perhaps due to the service or its VM crashing, the BigIP appliance detects this and then starts routing reader data to the slave xBRC. At that point, the slave becomes the new master and proceeds as described earlier in this section.

NOTES:   
  
The transition diagram erroneously suggests that some operations are synchronous that are, in truth, asynchronous. For example, the sending of DISCOVERY and synchronization messages is not performed *during* the processing of the incoming reader events. The events are received and sometime later the message sending occurs in an asynchronous fashion. The errors are a result of limitations of the software used to produce the diagram.  
  
Also note that if the master xBRC has its configuration changed after startup that the slave xBRC will detect the change and will re-synchronize its configuration.

## XBRMS HA Architecture

The xBRMS server uses a master/slave relationship similar to that used in xBRCs. However, the algorithm that determines which is master and which is slave uses a heartbeat mechanism where the heartbeat is stored in the xBRMS database. If the heartbeat from the master is not refreshed in time, the slave takes over. There is no state stored in the master xBRMS, so no transfer of information between master and slave is performed.

The BigIP is configured to monitor which xBRMS system reports as master, and directs requests accordingly.

The xBRMS UI systems use an active/active relationship for HA. No state information is exchanged between xBRMS UI systems as they operate independently. The BigIP is configured to handle xBRMS UI requests on a round robin basis.

## IDMS HA Architecture

IDMS uses an active/active relationship for HA. No state information is exchanged between IDMS systems as they operate independently. The BigIP is configured to handle IDMS requests on a round robin basis.

# Proper xBRC Configuration for HA

In order for HA to work properly it is essential that BigIP appliance and the master and slave xBRCs be properly configured. For example, if the slave xBRC is not properly receiving JMS messages from the master, it may remain in *unknown* state even after the master starts receiving reader events. This section identifies the key configuration settings that must be set for proper HA behavior.

## /etc/nge/config/environment.properties

It is essential that the JMS broker URL and credentials be configured. The parkid needs to be configured for the xBRC to be visible to the correct xBRMS, too:

* nge.eventserver.dataBrokerUrl
* nge.eventserver.xbrc.uid
* nge.eventserver.xbrc.pwd
* nge.xconnect.parkid

## xBRC Configuration Settings

It is also essential that HA is enabled and that the JMS topic and venue settings are correct:

* ESBInfo.jmstopic=XCONNECT.INTERACTIONS
* ControllerInfo.enableha=true
* ControllerInfo.venue=<non-default venue number>
* ControllerInfo.name=<non-default venue name>

By default, the venue name of an xBRC is “xBRC”. If the default is not changed, the xBRC will refuse to enable its HA functionality.

Some important notes:

1. The BigIP appliance must be properly configured to perform master-slave routing. For information on how to accomplish this please refer to the document, “*900-0174 Rev 1.6.1 xConnect BigIP HA Configuration*”.
2. It is essential that the master and slave xBRCs be using the same JMS broker (nge.eventserver.dataBrokerUrl), the same topic (ESBInfo.jmstopic) and the same venue (ControllerInfo.venue) and name (ControllerInfo.name).
3. It is essential that the JMS be working properly.
4. It is essential that the master and slave xBRCs have the same version of software installed and, in particular, be running the same “model”.
5. If any of these parameters are changed (or the ones in the previous section), the xBRC server **must be restarted**. It is not enough to set these values via xBRMS.

# Troubleshooting xBRC HA problems

The best tool for detecting xBRC HA problems is xBRMS. In xBRMS you can see the operating xBRCs, their venue settings and their HA state:

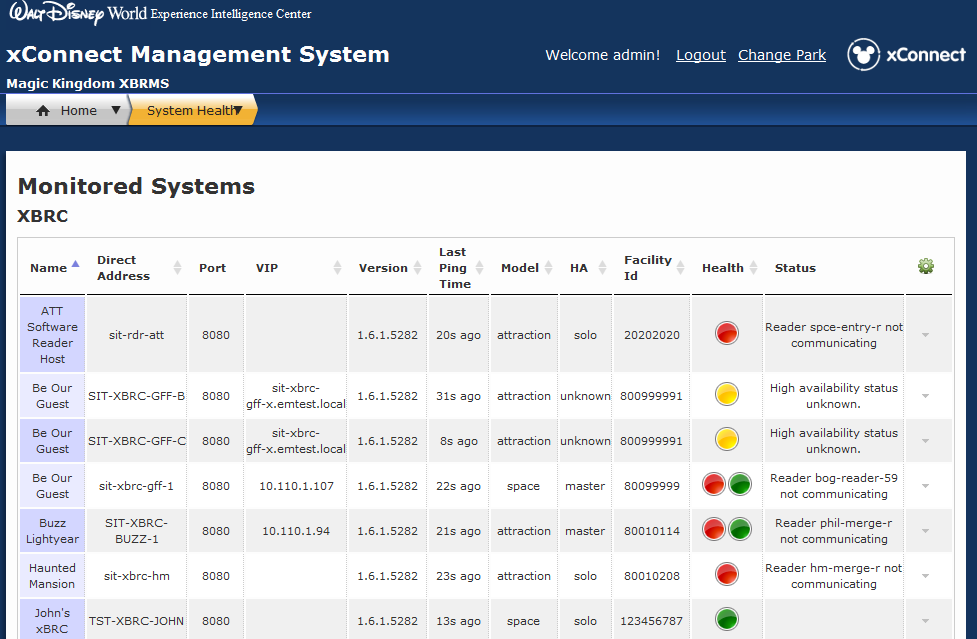


Figure - xBRMS System Health Status page

Here are some conditions that may arise and some suggestions as to how to correct them:

1. An xBRC shows its HA state as *solo*:  
   This indicates that the xBRC is not participating in HA. There are two causes for this: the *ControllerInfo.enableha* setting is not set to “true” or the xBRC venue name has not been set (it is defaulting to “xBRC”). Verify these settings through xBRMS, correct them as needed, then restart the affected xBRC.
2. The master xBRC shows its HA state as *unknown:*  
   This indicates that the xBRC is not receiving HELLO or data events from any readers. Here are some things to try:
   1. Verify that reader locations and readers have been properly configured on the xBRC.
   2. SSH to a reader and verify that it’s received proper configuration via DHCP telling it to communicate to the correct VIP for the xBRC. Verify that the reader can ping the xBRC and connect to its port 8080. Review the reader logs to see if it has had errors during its attempts to communicate with the xBRC.
   3. Run *tcpdump* on the xBRC to determine whether or not it’s receiving communication from its readers (it’s likely not).
3. The slave xBRC shows its HA state as *unknown* after the master enters *master* state:  
     
   The slave xBRC should change from *unknown* to *slave* state when it detects a JMS DISCOVERY message from the master that indicates that the master is now in *master* state. If the slave is not changing to slave state, it is because it is not receiving the appropriate DISCOVER message. Either the JMS configuration is incorrect, the JMS has failed or the master xBRC is having trouble sending messages to JMS. Verify the JMS settings for both xBRCs and verify that JMS is functioning properly.
4. Both the slave and master xBRCs are showing their HA states as *master*:  
     
   This can result from a couple of different situations:
   1. If the BigIP appliance is not properly configured, it may be load balancing the incoming reader events and sending them to both xBRCs. In this case their HA state will “bounce” as each xBRC will attempt to assert master state whenever it receives reader events.
   2. If the original master xBRC has failed and the slave has taken over, the original master will typically not be able to send out DISCOVERY messages and the xBRMS will show old information including the previous HA *master* state. The xBRMS, however, will typically flag the xBRC as non-responsive.
5. The slave xBRC does not function properly after taking over as master:  
     
   This can happen if the slave xBRC failed to properly synchronize its configuration with the master. Read through the */var/log/xbrc/xbrcController.log* on the slave xBRC and look for synchronization errors.

If none of these problems or solutions seem to be applicable, here are some other things to try:

* Rather than trusting the xBRMS, query the xBRCs directly by using a browser and checking their /status endpoints, for example:  
    
  <http://SOME-DIP-ADDRESS:8080/status>  
    
  The resulting XML will contain a tag for “hastatus”. Verify that the status is what you expect. Note that if the request fails, that indicates the xBRC is not operating properly and may need to be restarted.
* You can also use this technique to verify proper configuration settings:  
    
  <http://SOME-DIP-ADDRESS:8080/properties>  
    
  or  
    
  <http://SOME-DIP-ADDRESS:8080/currentconfiguration>  
    
  Both of these will return information about the property settings. The first form gives you a lot of metadata whereas the second form returns additional (non-property) information but in a more terse form.
* If you have access to the “JMS Client” tool, run this to monitor the DISCOVERY messages being sent by xBRCs. First, connect to the correct JMS broker and then establish a session to the right topic (typically, “XCONNECT.INTERACTIONS”). To minimize traffic, you can also enter a selector expression so that you only see JMS messages for the venue in which you are interested. For example:  
    
  xbrc\_facility=’XXXXXXX’ and xbrc\_type=’DISCOVERY’  
    
  As messages are displayed in JMS Client, click on them and inspect their content. Examine the XML paying particular attention to the “hastatus” tag.
* If xBRCs seem to be entering *master* state when they shouldn’t, make sure that the real master xBRC is running (so that the BigIP appliance doesn’t switch to the slave) and verify that the slave is not receiving any events or HELLO messages from readers. You can verify the former by viewing the contents of the */var/log/xbrc/eventdump.txt* file on the xBRCs. Verifying the latter (HELLO messages) is more difficult although clues may appear in */var/log/xbrc/xbrcController.log*.   
    
  If these log files aren’t useful, you can run *tcpdump* on the xBRCs and capture the network activity. Look for HTTP messages on port 8080.   
    
  Additionally, note that this problem frequently occurs when readers are configured to talk to xBRC DIPs instead of VIPs. Inspecting the *config.json, grover.conf* and *reader-cache* files in the readers can help to verify whether they are communicating using the correct address. This process also helps to assure that the DHCP system is properly connecting readers with their respective xBRCs.

# Troubleshooting xBRMS HA problems

At the time of this writing there are no known HA-related issues with xBRMS, and no troubleshooting outside of the BigIP configuration is available.

# Troubleshooting IDMS HA problems

At the time of this writing there are no known HA-related issues with IDMS, and no troubleshooting outside of the BigIP configuration is available.