Clearspan® Redundancy Guide

RELEASE 22

March 2018

Clearspan Redundancy Guide
Release #22
2744-008
March 2018

1 REVISION HISTORY

The following represents the revision history of this publication.

REVISION NUMBER	DATE COMPLETED	POINT OF CONTACT	DESCRIPTION
2744-008	2/2018	Clearspan Technical Publications	R22
2744-007	12/2016		R21
2744-006	09/2014		R20.0
2744-005	08/19/13	Bev Marsh – Aastra Technical Publications	R19.0
2744-004	08/15/13	Bev Marsh	R14.0
2744-003	11/12/08	Danielle Woelfle	Corrections and additions.
2744-002	07/23/08	Danielle Woelfle	Corrections and additions.
2744-001	03/06/08	Deb Bechtloff	Initial release of this publication.

2 SUMMARY OF CHANGES

This section describes the changes to this document for each release.

SUMMARY OF CHANGES FOR RELEASE 22.0

- Updated document for Release 22.0.
- Added section 6.1.1 FILE/ETC/HOSTS

SUMMARY OF CHANGES FOR RELEASE 21.0

- Updated document for Release 21.0.
- Added section 3.4.7 Media Server Failover Capability with a description of the Media Server Failover treatment.
- Updated server icons.

SUMMARY OF CHANGES FOR RELEASE 20.0

- Updated section 3.2.10 Xtended Services Platform.
- Added a description of the geo-redundancy proxy functionality.

SUMMARY OF CHANGES FOR RELEASE 19.0

- Updated section 3.2.10 Xtended Services Platform.

SUMMARY OF CHANGES FOR RELEASE 18.0

- Updated Figure 12 Redundancy Configuration.
- Updated section 5.3 Network Server Configuration.

SUMMARY OF CHANGES FOR RELEASE 17.0

- Updated section 5.3.2 Define Cluster of Application Servers and added section 5.4.3 Set Geo-Redundancy Parameters.
- Updated the maximum number of Network Servers in a cluster.
- Updated section 5.6.3 Monitor Data Replication.
- Removed references of mysql from Profile Server.
- Updated sections 3.4.1 Data Synchronization Between Cluster Peers.
- Updated section 3.2.1 Known Limitations.

- Updated section 5.4.6 SIP Configuration.

SUMMARY OF CHANGES FOR RELEASE 16.0

- Updated section 3.2.4 Network Server Cluster.

SUMMARY OF CHANGES FOR RELEASE 15.0

- Updated section 3.7.2 Application Server Failure: End User-Initiated Rollover.
- Added section 3.2.1 Known Limitation.
- Added a reference to the Xtended Services Platform (Xsp) redundancy model.
- Made a minor editorial change in section 3 Clearspan Redundancy Solution.
- Updated section 5 Redundancy Configuration.

3 INTRODUCTION

This document describes the configuration requirements to deploy a redundancy solution. It provides a high-level operating view, as well as, the configuration rules for Application Servers, Network Servers, Media Servers, Conferencing Servers, Element Management System servers, Profile Servers, Database Servers, Service Control Function Servers, and other network components.

4 CLEARSPAN REDUNDANCY SOLUTION

The Clearspan redundancy solution provides seamless, end-user transparency failover, in the event of an IP network outage or server failure. This solution ensures that no single point of failure results in a service outage.

Features of this solution are as follows:

- Automatic user rollover to a secondary Application Server when the primary Application Server fails to reply.
- Automatic tracking of the user's active Application Server through dynamic updates of the Network Server location database through the Application Server Redundancy (ASR) protocol.
- Access device-initiated and Application Server-initiated rollover.
- Automatic rollback of users' endpoints to the primary Application Server.
- Ability for the secondary Application Server to take the role of a SIP stateless proxy and relay SIP message between the primary Application Server and the SIP devices.
- Network Server support for multiple addresses per network device.
- No loss of active calls for a server-initiated rollover.
- Application Servers, Network Servers, Profile Servers (PS), Network Function Manager, or Element Management System (EMS) Servers in a cluster can be physically collocated (or not), allowing a geographic redundancy model to be implemented.
- The CommPilot login automatically selects the primary Application Server for an administrator or the active Application Server for a user.
- Conferencing servers that are deployed in stacks with a pair of front-end Conferencing Server–Application Servers are used to ensure availability in case of failure.
- EMS server supports an active-standby peering model.
- The Database Server (DBS) supports a primary-secondary peering model where read and write transactions can be performed on the primary, whereas the secondary only supports read transactions.

This redundancy solution is comprised of two major components: Hardware/IP redundancy (for more information, see section *Hardware/IP* Redundancy) and Clearspan redundancy (for more information, see section *Clearspan* Redundancy).

HARDWARE/IP REDUNDANCY

All Clearspan servers should be configured to use disk redundancy in RAID 1 mode. For more information, see the *Clearspan Software Management Guide*.

IP Network Multipathing provides your servers with the ability to recover from Layer 2 network failures (for example, NIC card or switch failures). If a failure occurs in a network adaptor, and you have an alternate adaptor connected to the same LAN, the system automatically switches all the network accesses from the failed adaptor to the alternate adaptor. This process ensures uninterrupted access to the network.

Disk Multipathing provides I/O load balancing and transparent failover to external mass storage devices, such as, a Fiber Channel or SCSI disk arrays. When configured properly, Disk Multipathing increases the overall availability of the system by eliminating failures at the direct-attach storage or SAN level.

4.1.1 LINUX

Clearspan recommends deploying Clearspan on IBM xSeries 336 or Blade Center, which provide a rugged, reliable, scalable, and highly available network environment.

4.1.1.1 Integrated RAID 1

IBM offers an integrated RAID 1 solution for the x336 and Blade Center series. It transparently maintains a mirror copy of data on another disk and in the event of a disk failure; it automatically uses the surviving copy.

4.1.1.2 Network Multipathing

Multipathing on Linux is achieved using the bounding feature. For configuration information, see the *Clearspan Software Management Guide*.

4.1.1.3 Disk Multipathing

Disk Multipathing on Linux is provided by the Linux Device-Mapper Multipath engine or by vendor-supplied multipathing solutions, such as, the IBM Storage Manager Linux RDAC multipath driver.

CLEARSPAN REDUNDANCY

Clearspan provides application-level redundancy on all Clearspan servers. Depending on the server's function, redundancy is provided through either clustering or pooling.

4.1.2 KNOWN LIMITATIONS

Simultaneous Calls from the Same Device on Different Application Servers

If a device is with an active call on one peer, then the device should always use the same peer if another call is performed while the first one is still active. Otherwise, using two Application Servers for two simultaneous calls has a limitation, such as *Call Transfer with Consultation*. This service does not work as the Application Server is trying to match a SIP-to-tag on the current peer. Because a second call was being performed on the other Application Server, then "no match found" and "SIP 603 Decline" messages are sent to the device for that SIP REFER.

Note that this limitation is alleviated for SIP calls when the geo-redundancy proxy is enabled since call attempts sent to the secondary server can be relayed to the primary server, allowing processing of all calls on the same application server.

Impact of Redundancy Solution on Trunking Call Capacity Management

Session context is not replicated between the primary and secondary Application Servers within a cluster. This can become apparent during Application Server rollover, rollback, or primary Application Server switching overload control. In such cases, a trunk group Call Admission Control (CAC) counter will have different values on the primary and secondary Application Server, each representing their own count of calls.

For example, when a rollover or switching overload control occurs, the trunk group CAC counter on the secondary Application Server is set to zero even if there are active calls for this trunk group on the primary Application Server. Also, if the primary Application Server is restarted before the rollback (for example, following a server failure), the trunk group CAC is reinitialized to zero on the primary Application Server, when trunk users are migrated back. The users involved in active calls remain managed by the secondary Application Server and they are not rolled back until the end of their calls. This may cause an overload during the transition period, until calls started on the primary server are completed.

4.1.3 BASIC DEFINITIONS

The following table provides a definition of the basic terms used to explain the Clearspan redundancy solution.

TERM	DEFINITION
Cluster	A cluster is a group of servers deployed in a data-sharing model. For the Clearspan redundancy model, the following clusters are possible. • One Network Server cluster for the entire network with two to twenty peers. • Multiple Application Server clusters, each with a primary and secondary node. Application Server clusters are added as required, with each cluster supporting up to 50,000 users.
Peer or node	A peer or node is a member of a cluster. It is not necessary for peers or nodes to be collocated.
Primary Application Server	One node of the Application Server cluster is defined as the primary node. Under normal operation, the primary node handles all user traffic. A secondary node handles user traffic in the event of failure on the primary node.
Active Application Server	The active Application Server is the Application Server node currently providing service for a given user on an endpoint-by-endpoint basis. One user can be active on the primary Application Server, while another is active on the secondary Application Server.
Rollover	A rollover is a condition whereby a user “fails over” to a secondary Application Server for service. A user failing over to a secondary Application Server, due to unavailability of the primary, is said to have rolled over to the secondary. A rollover only occurs from the primary Application Server to the secondary Application Server.

4.1.4 APPLICATION SERVER CLUSTER

Application Servers are deployed in primary/secondary cluster mode. Under normal operation, the primary Application Server handles all traffic. When the primary server or becomes unavailable, subscribers are automatically rolled over to the secondary Application Server on the next call (either originating or terminating) that involves a subscriber. This ensures continuous access to “dial tone” for subscribers.

The primary and secondary Application Servers are synchronized using the following mechanisms:

- **Cluster Data Replication**: The Application Server *TimesTen* database is replicated between the primary and secondary servers using *TimesTen* replication capabilities. Subscriber data-sharing ensures proper behavior upon rollover of an access device to the secondary Application Server.
- **Cluster File Synchronization**: Announcements, voice mail greetings, web branding, and other file-based configuration are synchronized using *RSYNC*, which is an open-source utility commonly used to mirror Internet web sites.

4.1.5 NETWORK SERVER CLUSTER

Network Servers are deployed in true cluster-server farm mode. A Network Server cluster can contain from two to twenty nodes. The primary/secondary concept does not exist in a Network Server cluster. Any Network Server in a cluster can be used to process a call in a load-balancing or fixed-order fashion. When there is a network or server failure, any Network Server in the cluster can take over.

The Network Server cluster is synchronized using the following mechanisms:

- **Cluster Data Replication**: The Network Server *TimesTen* database is replicated across the cluster using *TimesTen* replication capabilities.
- **Cluster File Synchronization**: Web branding and other file-based configuration is synchronized using RSYNC.

4.1.6 MEDIA SERVER POOLING

Media Server redundancy is accomplished through N+1 pooling. All Media Servers are independent. A failure of a Media Server results in the Application Server timing out on attempts to reach that server, and then routes to the next available Media Server.

4.1.7 PROFILE SERVER CLUSTER

Profile Servers are deployed in true cluster-server farm mode. The primary/secondary concept does not exist in a Profile Server cluster. Any Profile Server in a cluster can be used to process a request in a load-balancing or fixed-order fashion. When there is a network or server failure, any Profile Server in the cluster can take over.

The Profile Server cluster is synchronized using the **Cluster File Synchronization**. Using this all the device file resources and file-based configuration are synchronized using RSYNC.

4.1.8 CALL DETAIL SERVER

The Call Detail Server is not redundant.

4.1.9 XTENDED SERVICES PLATFORM

Each individual Xtended Services Platform (Xsp) is not redundant but operates in a stateless farm, which means that any Xtended Services Platform in a farm can process any request. For example, a provider can create a cluster fully qualified domain name (FQDN) representing the Xtended Services Platform farm. This FQDN resolves to all of the Xtended Services Platforms in a round-robin fashion where any of the Xtended Services Platforms can handle any request from the Clearspan Xtended Services Interface (Xsi), Device Management on Clearspan, and so on.

The following is a typical configuration example describing a farm with three Xtended Services Platforms:

On XSP1:

```
XSP_CLI/Interface/Http/HttpServer> get

  Interface    Port                    Name   Secure        Cluster FQDN
=========================================================================
192.168.1.1     443    xsp1.company.com     true    web.company.com
192.168.1.2      80    xsp1.company.com    false    web.company.com
```

On XSP2:

```
XSP_CLI/Interface/Http/HttpServer> get

 Interface     Port                    Name   Secure         Cluster FQDN
=========================================================================
 10.10.1.1      443    xsp2.company.com     true    web.company.com
 10.10.1.2       80    xsp2.company.com    false    web.company.com
```

On XSP3:

```
XSP_CLI/Interface/Http/HttpServer> get

 Interface     Port                    Name   Secure        Cluster FQDN
=========================================================================
 11.11.1.1      443    xsp3.company.com     true    web.company.com
 11.11.1.2       80    xsp3.company.com    false    web.company.com
```

In addition, an HTTP alias can be defined if required. The alias my3.company.com would be served by the xsp3.company.com server.

On XSP3:

```
XSP_CLI/Interface/Http/HttpAlias> get

 Interface                  Name                     Cluster FQDN
=================================================================
 11.11.1.1        my3.company.com              my.company.com
```

4.1.10 DATABASE SERVER

Database Server (DBS) redundancy is supported at two different levels, locally and geographically. Local redundancy is achieved using clustering techniques (N+1), where each node shares a common database and service client requests using load balancing algorithms. Geographic site redundancy is achieved using a primary/standby model (2N), where client requests are processed by the primary site and replicated to the standby site.

For more information, see the Clearspan Database Server Configuration Guide.

4.1.11 SERVICE CONTROL FUNCTION SERVER

Service Control Function Servers are deployed in an active-active peering model for Signaling System 7 (SS7) and SIP protocols, any of the Service Control Function Servers can handle the calls. On the SS7, the network-side redundancy is done via Stream Control Transmission Protocol (SCTP) Multi-Homing as follows:

- Multi-Link/Multi-IP address eliminates single point of failure
- Multi-stream reliable message delivery
- Uses more than one IP network interface for true fault tolerance
- Monitors endpoints for availability

For more information, see the Clearspan Service Control Function Configuration Guide.

4.1.12 NETWORK FUNCTION MANAGER

The Network Function Managers are deployed in an active-active cluster-server farm mode. Any Network Function Manager in a cluster can be used to process a request in a load-balancing or fixed-order fashion.

The Network Function Managers are synchronized using two mechanisms:

- Database replication: All database write transactions are replicated on all members of the cluster to ensure consistency across the cluster.
- Configuration Replication: Some configuration elements of the Network Function Manager are replicated from a configuration master (replicated) to configuration slave (replica).

For more information, see the Clearspan Network Function Manager Configuration Guide.

REDUNDANCY PROTOCOLS AND UTILITIES

The Clearspan redundancy solution makes use of the following protocols and utilities. *Figure 1* Protocols and Utilities illustrates how these protocols and utilities are used in the Clearspan redundancy solution.

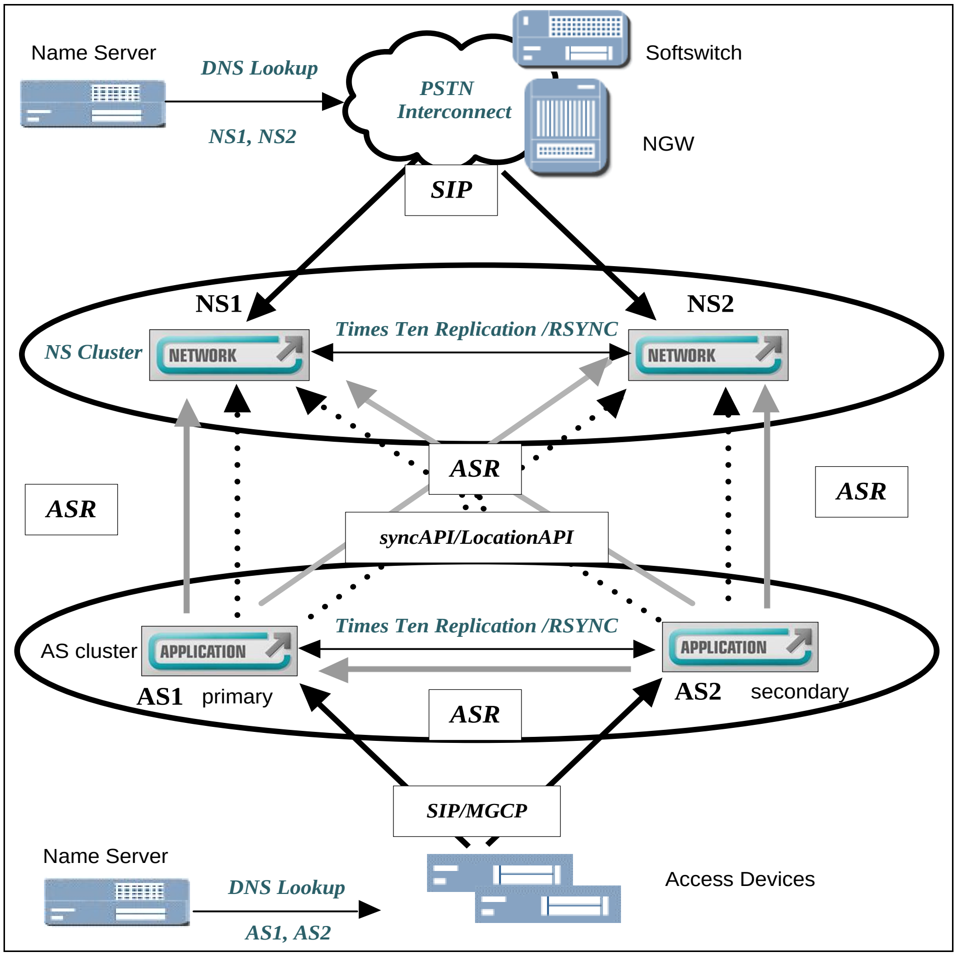

Figure 1 Protocols and Utilities

- **DNS (Domain Name Server) – Access Side**: Access devices perform DNS lookups on the Application Server cluster Fully Qualified Domain Name (FQDN). The DNS response always returns the primary, followed by the secondary. The access device tries the primary Application Server, and if it does not respond, it advances to the secondary Application Server.
- **DNS – Network Side**: Network devices perform DNS lookups on the Network Server cluster FQDN. The DNS response may return to cluster peers in any order. A network device tries the first Network Server returned, and if it does not respond, it advances to the next Network Server.
- **SyncAPI**: This Clearspan proprietary protocol is used to automatically push groups and users from the Application Server to the Network Server.
- **LocationAPI**: This is a Clearspan proprietary protocol. Application Servers query the Network Server for primary or secondary Application Server HTTP access information (for example, node FQDN).
- **ASR (Application Server Redundancy)**: This Clearspan proprietary protocol is used by an Application Server to inform a Network Server that a user has changed their active Application Server. For example, it is used when a user has rolled over to the secondary Application Server or a user has rolled back to the primary Application

Server. This protocol is also used by the secondary Application Server to inform the primary Application Server that a number of users must be rolled back to the primary.

- **TTRep (TimesTen Replication)**: This is a database replication daemon used to automatically replicate database transactions to other cluster member servers.
- **RSYNC**: This is a third-party open source utility used to synchronize files between cluster members.
- **Geo-Redundancy Proxy**: This is the ability for the secondary Application Server to act as a SIP stateless proxy allowing SIP messages to be relayed between the primary Application Server and the SIP device.
- **SIP Rollback Confirmation**: An end-user confirmation might be required when a user rolls back to the primary Application Server. For SIP users, an *OPTIONS* message is sent to the endpoint. Users are not rolled back until the endpoint responds to the message.

REDUNDANCY KEY CAPABILITIES

The Clearspan redundancy solution is composed of the following capabilities:

- Data synchronization between cluster peers
- Application Server user rollover capability
- Network Server failover capability
- Subscriber's active Application Server tracking
- Application Server user rollback capability
- Application Server geo-redundancy proxy capability

4.1.13 DATA SYNCHRONIZATION BETWEEN CLUSTER PEERS

The Application Server and Network Server databases are replicated between all respective members of the cluster using TimesTen Replication. The EMS database is replicated between peers using mySql replication. Required files are synchronized between peers using RSYNC.

4.1.14 APPLICATION SERVER USER ROLLOVER CAPABILITY

SIP access devices roll over to an alternate call agent/proxy (secondary Application Server) in the event of primary server failure. Rollover is on an endpoint-by-endpoint basis. Upon receiving an originating or terminating call request involving a user, the secondary Application Server verifies the active node of the user. If the user's active node is the primary Application Server, the secondary Application Server takes ownership of that endpoint and informs the Network Server cluster of the active node change for that user. The secondary Application Server remains the active node for that user until the user is rolled back to the primary Application Server.

A user can be rolled over in one of three ways:

- **User-initiated rollover**: The user originates a call that times out on the primary Application Server, thereby causing the user to roll over to the secondary Application Server.
- **PSTN-initiated rollover**: The user's incoming PSTN calls time out on the primary Application Server, thereby causing the user to roll over to the secondary Application Server.
- **CommPilot-initiated rollover**: When the primary Application Server is out of service and the user logs into the CommPilot web interface on the secondary Application Server, the user rolls over to the secondary Application Server.

Both the primary and secondary Application Server addresses must be either identified, on the access device or mapped to a common domain name (when DNS is used and supported by the access device). The domain name server should always return the Application Server cluster in a fixed order (for example, primary Application Server followed by secondary Application Server).

Note: A user initiated rollover and PSTN initiated rollover might not occur under certain conditions if the geo-redundancy proxy is enabled. If for instance, the primary Application Server remains reachable from the secondary, the messages can be relayed to the primary, preventing the need to roll over, as explained in section 3.7 Geo-Redundancy Proxy Behavior.

4.1.15 NETWORK SERVER FAILOVER CAPABILITY

The Clearspan redundancy solution makes use of a network device's capability to roll over to an alternate SIP proxy when a failure occurs. The Network Server cluster member addresses must be mapped to a common DNS domain name. The network device performs a DNS lookup on the Network Server cluster name. The DNS name server can return to the Network Server cluster peers in any order.

4.1.16 SUBSCRIBER'S ACTIVE APPLICATION SERVER TRACKING

The Network Server tracks a subscriber's active Application Server (location). Tracking is achieved through a registration protocol between the application and the Network Server that is used to exchange the subscriber's active Application Server node.

When a call arrives at the Network Server, the Network Server always returns the addresses of the primary and secondary Application Servers (in the 302 contact list). The returned contact list always has the active Application Server for a subscriber returned first (that is, with highest q-value).

4.1.17 APPLICATION SERVER USER ROLLBACK CAPABILITY

Under normal circumstances, the user should always be active on the primary Application Server. A user that has been rolled over is rolled back to the primary Application Server once the Application Server is back in service.

Users can be rolled back to the primary Application Server in one of two ways:

- **Application Server Redundancy (ASR)-initiated rollback**: The expiration of the ASR timer on the secondary Application Server triggers an ASR rollback message to be sent to the primary Application Server, identifying all users currently active on the secondary Application Server. If the primary is available, all users listed in the ASR rollback message are rolled back to the primary Application Server (that is the primary Application Server becomes the active node for the users). Users involved in active calls are not included in the ASR rollback message. The ASR timer default is 15 minutes. Note that the rollback is triggered only after an endpoint confirmation (SIP users are sent an OPTIONS message. However, no OPTIONS messages are sent to SIP devices when the geo-redundancy proxy is enabled.
- **User-initiated rollback**: A user may trigger a rollback by initiating a call that is processed by the primary Application Server. Upon receiving the call request from that user, the primary Application Server verifies the user's active node. If the user is currently active on the secondary Application Server, the user is rolled back to the primary Application Server.

4.1.18 APPLICATION SERVER GEO-REDUNDANCY PROXY CAPABILITY

SIP devices can send messages to the secondary Application Server when the primary Application Server is unreachable or slow to respond, or for other various reasons and this, even if the primary server is in fact available and healthy. In such cases, upon receiving SIP messages, the secondary Application Server can be configured to act as a SIP stateless proxy and as a result can relay messages between the device and the primary Application Server. This prevents users becoming active on the secondary server and allows the primary server to process all calls for a user locally, eliminating limitations caused by distributing call processing over multiple servers.

Likewise, for call termination, when the primary Application Server considers a user unreachable, it can send outbound messages to the proxy on the secondary Application Server. This allows the primary to reach users that might otherwise be unreachable due to network conditions.

This proxy server behavior is a configuration option that may be enabled or disabled.

If the geo-redundancy proxy is disabled, then the secondary Application Server never relays received SIP messages and instead can rollover the user as described in section *3.4.2 Application Server User Rollover Capability* when processing incoming request from a device. Similarly, when the feature is disabled, the primary Application Server does not terminate calls by going through the secondary Application Server proxy and instead attempts to send the request directly to the user's device. For more information on the geo-redundancy proxy, see section *3.7 Geo-Redundancy Proxy Behavior*.

4.1.19 MEDIA SERVER FAILOVER CAPABILITY

If the primary Application Server becomes unreachable while the Media Server is playing an announcement, the Media Server sends SIP messages to the secondary Application Server.

If session data replication is disabled, the secondary Application Server is unaware of the call session information and sends a new announcement request to the Media Server. The Media Server then plays an announcement that informs the user that the Application Server is experiencing system difficulties. The user is then asked to hang up and try again later.

If session data replication is enabled, this mechanism is not required as the secondary Application Server is able to resume the call in progress.

Note: This mechanism is disabled by default and must be enabled by setting the *bw.sip.playsystemdifficultiestreatment* execution container option to *true*. If the container option is not defined or set to *false*, the Media Server does not play the system difficulties treatment.

ACTIVE CALL FAILOVER BEHAVIOR

The Clearspan redundancy solution does not replicate active call states across Application Server peers. When there is a server failure, active calls that were established on the failed Application Server are taken down. Media streams negotiated between the active endpoints during the call initiation are independent of the Application Server and continue uninterrupted.

Active calls initiated on a failed server, have special behaviors with respect to Call detail record (CDR) generation and mid-call event handling. The secondary Application Server normally handles all subsequent calls made after the termination of the failed-over active call.

4.1.20 FAILOVER CDR GENERATION

Upon termination of the active call, the terminating endpoint attempts to inform the failed server of the termination event (for example, a SIP BYE message). Since the failed server does not respond to the termination event, the endpoint then sends the termination event to the other Application Server peer.

Upon receiving the termination event for the call leg that was not initiated on that server, the server generates a failover CDR containing a *PartialCallEnd* module. This CDR can be correlated to the CDR that was generated on the failed server.

4.1.21 MID-CALL EVENT HANDLING

Active calls that were started on the failed server remain up. Depending on the device type (SIP), mid-call events are handled in a different manner.

4.1.21.1 SIP Hold/New Call Initiation

A SIP user initiating a phone HOLD condition results in an INVITE with SDP 0.0.0.0 being sent to the failed server. The message timeouts on the failed server and is sent to the secondary or backup server. The secondary server returns a SIP "481 Call Leg Does Not Exist" message causing the phone to take down the call.

4.1.21.2 SIP Endpoint Call Termination

An incoming call for a SIP user is routed through the secondary server. The secondary server is not aware of the existing call and sends an INVITE message to the SIP endpoint. The endpoint displays the new call. If the user answers the incoming call, the existing call is dropped since the existing call is put on hold. This results in an INVITE with SDP 0.0.0.0 timing out the failed server and being sent to the secondary server. The secondary server returns a SIP "481 Call Leg Does Not Exist" message, causing the phone to take down the call.

The secondary server handles the answered incoming call and all possible subsequent call events normally.

GEO-REDUNDANCY PROXY BEHAVIOR

The secondary Application Server can take the role of SIP stateless proxy and, under some conditions, relay SIP messages between the primary Application Server and the SIP devices. When doing so, the proxy modifies the SIP messages to help ensure that it remains in the signaling path for subsequent responses and mid-dialog requests. This is achieved through the values supplied within the Via and Record-Route headers. The following provides some details on the routing mechanism used.

4.1.22 CALL ORIGINATION

Figure 2 illustrates a scenario where a call originated from a user is relayed by the secondary Application Server proxy.

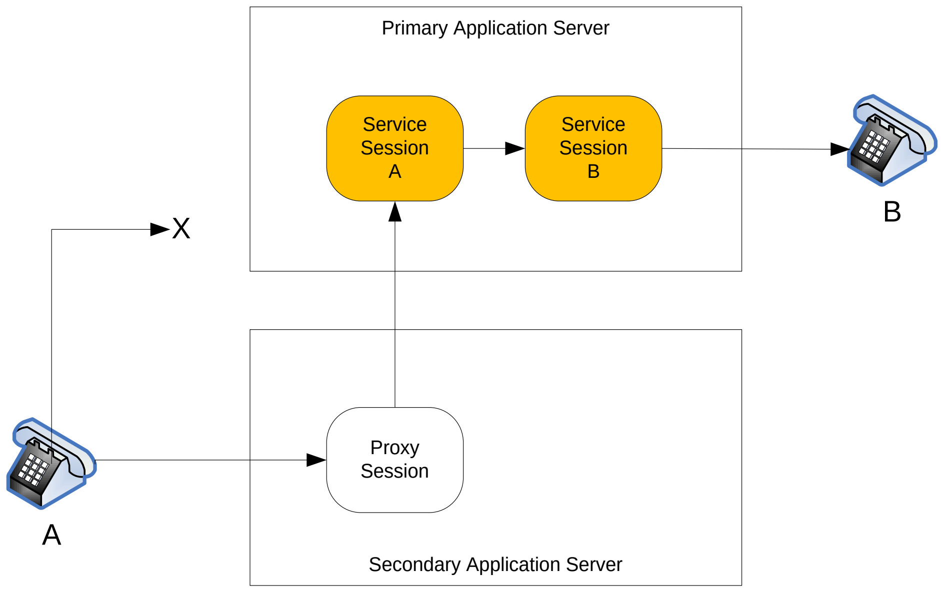

Figure 2 Call Origination Relayed by Secondary Application Server Proxy

In this scenario, the phone labeled "A" is an access device, which first attempts unsuccessfully to send an INVITE request to the primary Application Server. The access device eventually fails over to the secondary Application Server. Upon receiving the SIP request, the secondary Application Server relays the INVITE request to the primary Application Server since the user has no local active call and the primary Application Server is available for processing. In this way, the new call proceeds without a loss of functionality, despite a loss of network connectivity between the access device and the primary Application Server.

In the described scenario, the secondary Application Server made the decision to route the call as a proxy based on the availability of the primary Application Server. Availability information about the peer Application Server is obtained by exchanging information over a dedicated communication link between the servers. The secondary Application Server therefore knows if the primary Application Server is in a state suitable for processing an incoming call or not. A server is identified as unavailable if, for example, it is in an overload state, or in locked state, or simply unreachable.

In general, the secondary Application Server relays new initial SIP INVITE and SIP SUBSCRIBE request to the primary if:

The primary Application Server is available and,

The user has no local active call on the secondary Application Server.

4.1.23 CALL TERMINATION

Figure 3 illustrates a scenario where a call termination is relayed by the secondary Application Server proxy.

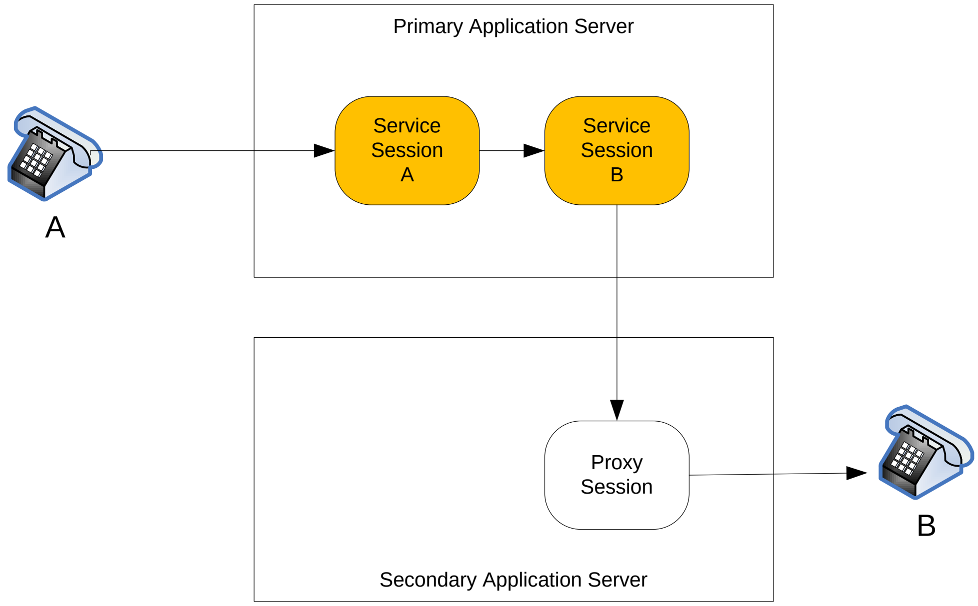

Figure 3 Call Termination Relayed by Secondary Application Server Proxy

In this scenario, phone A sends an INVITE request to the primary Application Server for phone B. The primary Application Server considers, based on an internal indicator, that the device endpoint at phone B is unreachable. The primary Application Server also knows that the secondary Application Server is available to act as a proxy server. As a result, the primary Application Server sends the INVITE request to the secondary Application Server, which performs the role of a proxy server and routes the INVITE request to phone B.

In the described scenario, the primary Application Server routing decision was based on a priori knowledge about the device endpoint reachability and on the secondary server availability. This knowledge about the device endpoint reachability is controlled by an internal isReachableFromPrimary indicator that is associated with the device endpoint. The primary Application Server sets this indicator to “false” when it receives an initial INVITE or SUBSCRIBE from the secondary Application Server.

The primary Application server sends a new initial SIP INVITE to the secondary Application Server proxy if:

- The secondary Application Server is available and,
- The isReachableFromPrimary indicator is set to “false” for any of the user’s endpoint.

In other cases, the SIP INVITE is sent directly to the access device.

Note: This behavior may also apply when sending SIP OPTIONS, SIP NOTIFY, and SIP MESSAGE, depending on the exact context. Note also that the isReachableFromPrimary indicator is maintained only for an access device. No indicator is maintained for network devices like network gateways for instance.

The isReachableFromPrimary indicator is reset to “true” in one of two ways:

Application Server connectivity test: The primary Application Server periodically checks if the connectivity with the device endpoint has been reestablished by sending a SIP OPTIONS messages to the device. If a SIP OPTIONS response is successfully received from the device, then the isReachableFromPrimary indicator is set to “true”, allowing future traffic to go directly to the device endpoint instead of transiting through the proxy. The interval between connectivity tests is the same as interval used by the ASR timer described in section 3.4.5 Application Server User Rollback Capability.

User-initiated activity: The isReachableFromPrimary indicator is set to “true” when receiving an initial SIP INVITE or SUBSCRIBE directly from the user’s device endpoint.

4.1.24 PEER SIP CONNECTIVITY MONITORING

Before the secondary Application Server routes an initial INVITE request to the primary Application Server (call origination scenario), the secondary Application Server must know that it has connectivity to the primary Application Server. Likewise, before the primary Application Server routes an initial INVITE request to the secondary Application Server (call termination scenario), it must know that it has connectivity to the secondary Application Server. To maintain this connectivity awareness, SIP connectivity monitoring could be used.

When SIP connectivity monitoring is enabled, the monitoring Application Server regularly sends an OPTIONS request to its peer Application Server. The interval between OPTIONS requests is configurable via a system parameter. After the monitoring Application Server sends the OPTIONS request, it sets a timer, also controlled by a system parameter, and waits to receive a response. If it receives a response before the timer expires, it considers its peer Application Server to be reachable. Otherwise, when the timer expires, it considers the peer unreachable.

SIP connectivity monitoring is disabled by default. If the Application Server uses the same network interface for SIP messages as for the redundancy link, then the monitoring of the redundancy link is sufficient, and separate monitoring is unnecessary. However, SIP connectivity monitoring is needed when the Application Servers use a different network interface for SIP messages and for the redundancy link.

4.1.25 REQUIRED NETWORK SERVER POLICY

The ability for the secondary Application Server to act as a proxy ensures that a maximum level of traffic is processed on the primary Application Server. However, in some cases, even if this feature is enabled, the secondary Application Server may decide not to relay SIP messages to the primary. In such cases, the SIP messages routed to the secondary Application Server are processed locally and it could result in the user rollover.

Under these conditions, even though the user has some active call(s) on the secondary, new SIP requests sent by a user’s device to the primary are not relayed by the primary Application Server to the secondary. Instead, these new messages are processed locally on the primary Application Server. This can cause a user to have active calls on both the primary and secondary Application Servers at the same time.

This situation can be avoided by configuring the Network Server with the OrigRedirect policy. This way, new SIP requests originated by Clearspan users through a Session Border Controller (SBC) are sent to the Network Server, which can redirect the request to the appropriate Application Server based on where a user is hosted. This eliminates the need for the primary Application Server to relay incoming requests to the secondary Application Server when a user has active calls on the secondary Application Server since these requests are redirected to the secondary Application Server by the Network Server.

CLEARSPAN REDUNDANCY WALKTHROUGH

This section provides examples of usual redundancy failures and recovery situations. The following examples are provided:

- Network Server failure: Route advance to next Network Server
- Application Server failure: End user-initiated rollover
- Application Server failure: PSTN-initiated rollover
- Application Server failure: CommPilot-initiated rollover
- Application Server recovery: Application Server-initiated rollback
- Application Server recovery: User-initiated rollback
- Application Server geo-redundancy proxy: Call origination
- Application Server geo-redundancy proxy: Call termination

A step-by-step walkthrough is provided for each example. The end-user device is SIP-enabled for each example. Messages to the Network Server cluster are shown going to NS1 and NS2 in a load-balancing fashion.

4.1.26 NETWORK SERVER FAILURE: ROUTE ADVANCE TO NEXT NETWORK SERVER

The scenario for this failure is that there is an incoming PSTN call to a User 1, who is hosted on a redundant Application Server cluster. The Network Server NS1, which is the first Network Server returned in the DNS response, is out of service.

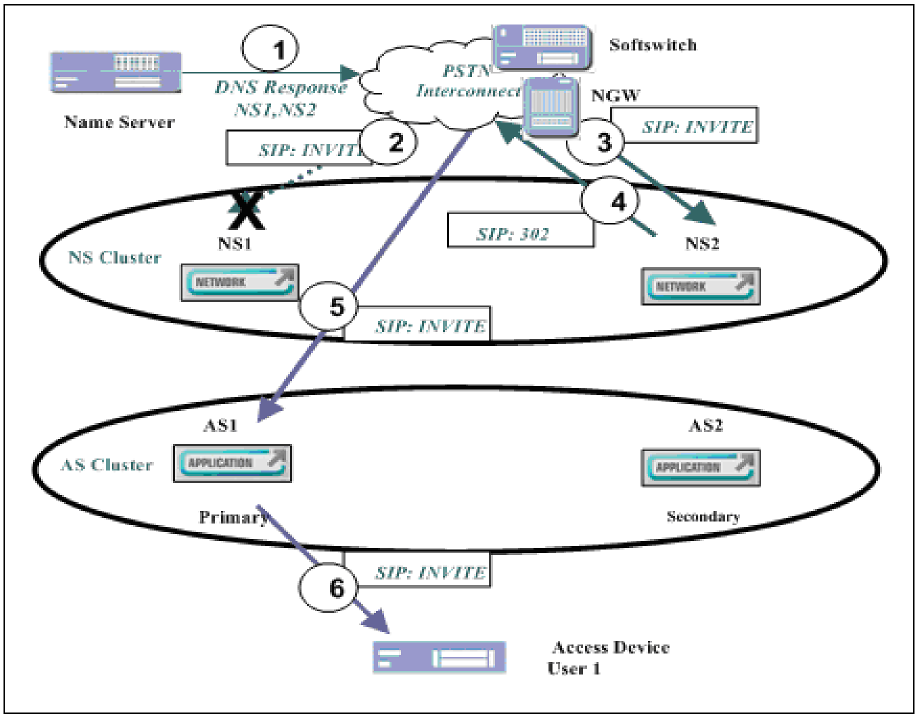

Figure 4 Network Server Failure: Route Advance to Next Network Server

The sequence of events is as follows:

1. The network device performs a DNS lookup on the Network Server cluster FQDN. The name server returns NS1 followed by NS2 in the response.
2. The network device sends an INVITE to NS1. NS1 is out of service and fails to respond.
3. The network device times out on NS1 and route advances to NS2. The network device sends an INVITE to NS2.
4. NS2 returns a 302 TEMPORARILY MOVED with a contact list equal to AS1 followed by AS2.
5. The network device sends an INVITE to AS1.
6. AS1 processes the incoming INVITE and terminates the call to User 1.

4.1.27 APPLICATION SERVER FAILURE: END USER-INITIATED ROLLOVER

The scenario for this failure is that User 1 initiates a call to a PSTN subscriber and the primary Application Server is out of service.

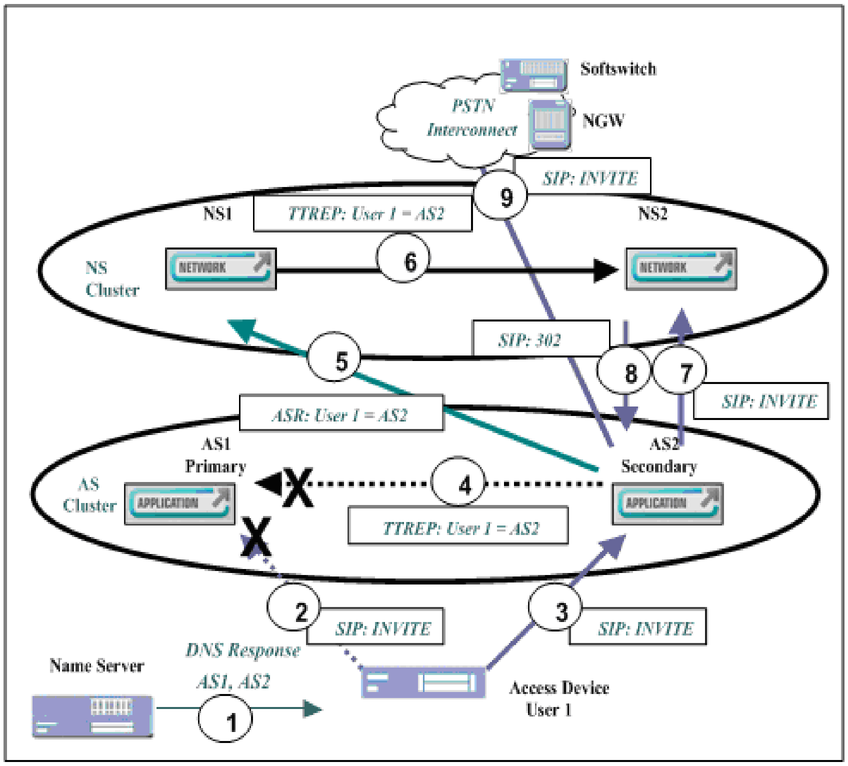

Figure 5 Application Server Failure: End User-Initiated Rollover

The sequence of events is as follows:

1. The access device performs a DNS lookup on the Application Server cluster FQDN. The name server returns AS1 followed by AS2 in the response.
2. The access device sends an INVITE to AS1. AS1 is out of service and fails to respond.
3. The access device times out on AS1 and route advances to AS2. The access device sends an INVITE to AS2.
4. AS2 receives the INVITE. AS2 has detected that AS1 is unable and decides to process the message locally and to roll over the user. AS2 updates its database to indicate that User 1 is now active on AS2. Since AS1 is not available, the transaction goes to the AS2 replication buffer (to be pushed to AS1 when it recovers).
5. AS2 informs the Network Server cluster that User 1 is now active on AS2 by sending an ASR message.

6. NS1 updates the database for User 1 to indicate that AS2 should be the first contact. NS1 informs NS2 via TTREP that User 1 is now active on AS2. All calls destined to User 1 cause the Network Server cluster to return a contact with AS2 followed by AS1.
7. AS2 sends an INVITE to the Network Server cluster.
8. NS2 returns a 302 TEMPORARILY MOVED with a contact equal to the NGW.
9. AS2 sends an INVITE to the PSTN NGW.

Note: The same scenario occurs if User 1 tries to subscribe to any of the SIP access event packages (for example, as-feature-event, clearspan-call-center-status, dialog, and so on) via a SIP SUBSCRIBE message. If User 1 sends the SIP SUBSCRIBE directly to AS2, and if AS2 decides to process the request locally, then AS2 migrates the user. This user migration allows the secondary Application Server to send NOTIFY messages. Note that this behavior does not apply to the SIP line-seize event package. For the line-seize event package, no user migration is performed when AS2 receives a SUBSCRIBE.

4.1.28 APPLICATION SERVER FAILURE: PSTN-INITIATED ROLLOVER

The scenario for this failure is that there is an incoming PSTN call for User 1 and the primary Application Server is out of service.

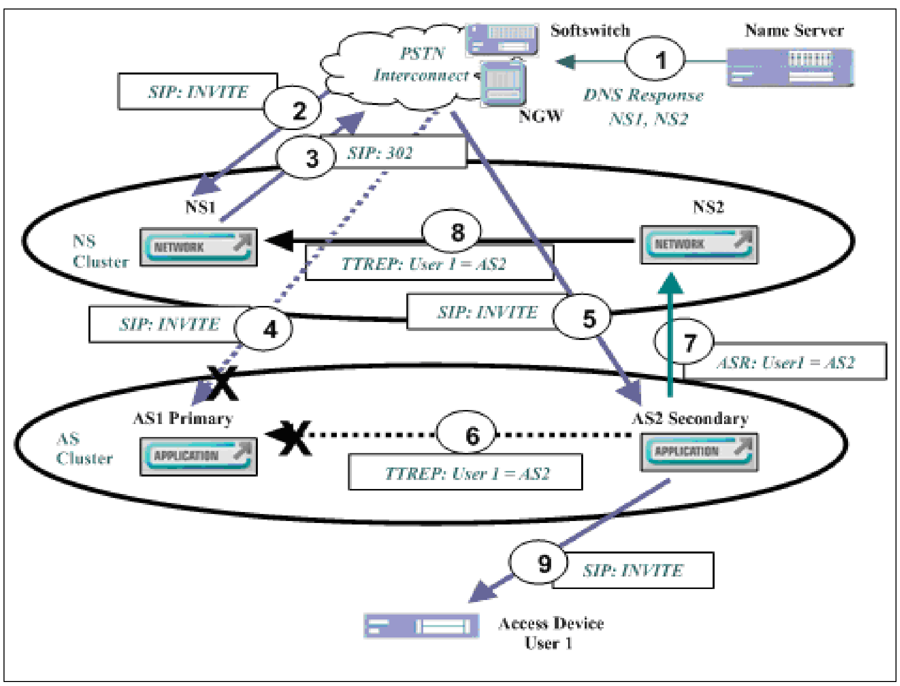

Figure 6 Application Server Failure: PSTN-Initiated Rollover

The sequence of events is as follows:

1. The network device performs a DNS lookup on the Network Server cluster FQDN. The name server returns NS1 followed by NS2 in the response.

2. The network device sends an INVITE destined for User 1 to NS1.
3. The network device returns a 302 TEMPORARILY MOVED message, indicating that User 1 is located on AS1 followed by AS2.
4. The network device sends an INVITE to AS1. AS1 is out of service and does not respond.
5. The network device times out on AS1 and route advances to AS2. The network device sends an INVITE to AS2.
6. AS2 receives the INVITE. AS2 has detected that AS1 is unavailable and decides to process the message locally and to roll over the user. AS2 updates its database to indicate that User 1 is now active on AS2. Since AS1 is not available, the transaction goes to the AS2 replication buffer (to be pushed to AS1 when it recovers).
7. AS2 informs the Network Server cluster that User 1 is now active on AS2, by sending an ASR message.
8. NS1 updates the database for User 1 to indicate that AS2 should be the first contact. NS1 informs NS2 via TTREP that User 1 is now active on AS2. All calls destined to User 1 cause the Network Server cluster to return a contact with AS2 followed by AS1.
9. AS2 processes the call and sends an INVITE to the User 1 access device.

4.1.29 APPLICATION SERVER FAILURE: COMMPILOT-INITIATED ROLLOVER

The scenario for this failure is that the primary Application Server is out of service and User 1 attempts to log in to CommPilot before having originated or terminated a call.

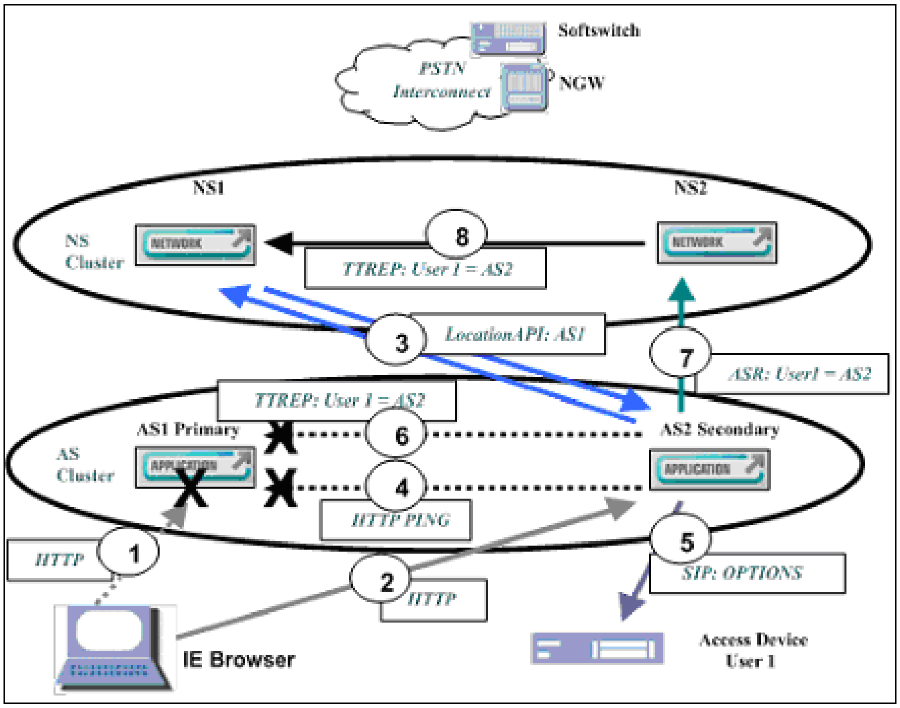

Figure 7 Application Server Failure: CommPilot-Initiated Rollover

The sequence of events is as follows:

1. User 1 attempts to log in to CommPilot via an IE browser. The browser tries AS1, which is out of service.
2. The browser times out on AS1 and tries AS2. AS2 processes the HTTP request, allowing User 1 to log in to the portal.
3. AS2 checks the database and discovers that User 1 is active on AS1. User 1 should log in to AS1 since it is the active server for that user. AS2 sends a LocationAPI lookup to the Network Server cluster to obtain the access address of AS1.
4. AS2 performs an HTTP ping to the access address of AS1 to see if the login attempt can be redirected to AS1. AS1 does not respond.
5. Since AS1 does not respond to the HTTP ping, AS2 rolls over the user to the secondary Application Server. A SIP OPTIONS message is sent to User 1's access device.
6. Upon receiving a response to the OPTIONS message, AS2 updates its database to indicate that User 1 is now active on AS2. AS2 attempts to inform AS1 via TTREP that User 1 has migrated to AS2. AS1 does not respond, so the transaction goes to the AS2 replication buffer.
7. AS2 informs the Network Server cluster that User 1 is now active on AS2 by sending an ASR message.

8. NS1 updates the database for User 1 to indicate that AS2 should be the first contact. NS1 informs NS2 via TTREP that User 1 is now active on AS2. All calls destined for User 1 cause the Network Server cluster to return a contact with AS2 followed by AS1.

4.1.30 APPLICATION SERVER RECOVERY: APPLICATION SERVER-INITIATED ROLLBACK

The scenario for this failure is that the primary Application Server has been returned to service. User 1 is automatically rolled back to the primary Application Server AS1 upon expiration of the ASR timer on the secondary Application Server.

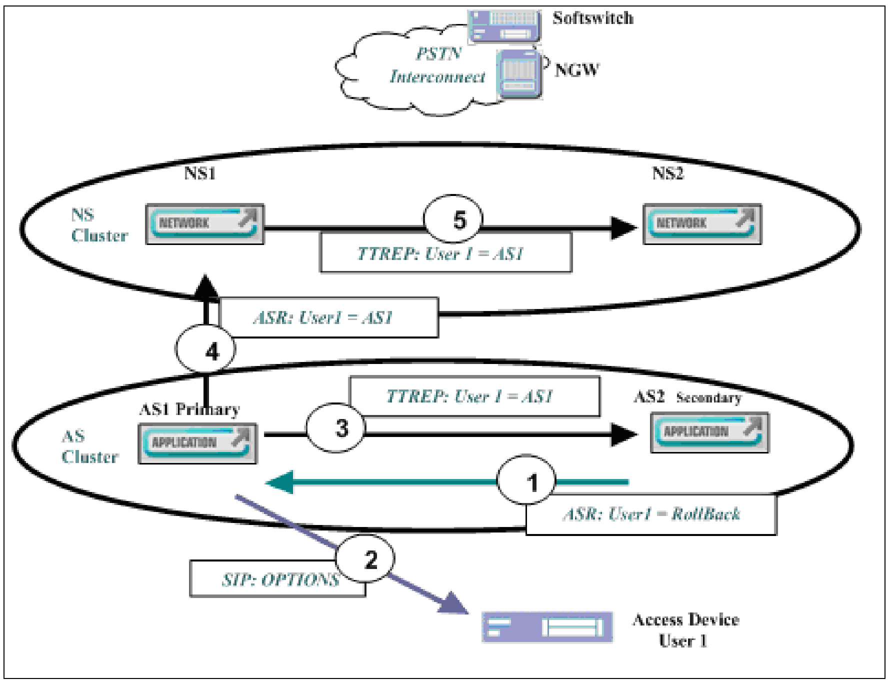

Figure 8 Application Server Recovery: Application Server-Initiated Rollback

The sequence of events is as follows:

1. The ASR redundancy timer expires on AS2, triggering an ASR message to be sent to AS1. The ASR message lists the users that are currently active on AS2.
2. AS1 processes the ASR rollback request sending rollback confirmation messages (SIP OPTIONS) to User 1's access device.
3. Upon receiving a response to the OPTIONS message, AS1 initiates a user rollback by updating the active node of User 1 to the primary Application Server AS1, and informing the secondary Application Server of the active node change via TTREP.
4. The primary Application Server AS1 informs the Network Server cluster of the user active node change, by sending an ASR message to one of the Network Servers in the cluster.

5. NS1 updates the database for User 1 to indicate that AS1 should again be the first contact. NS1 informs NS2 via TTREP that User 1 is now active on AS1. All calls destined to User 1 cause the Network Server cluster to return a contact with AS1 followed by AS2.

4.1.31 APPLICATION SERVER RECOVERY: USER-INITIATED ROLLBACK

The scenario for this failure is that the primary Application Server has been returned to service. User 1 initiates a call to a subscriber in the PSTN that goes to the primary Application Server. User 1 is rolled back to the primary Application Server.

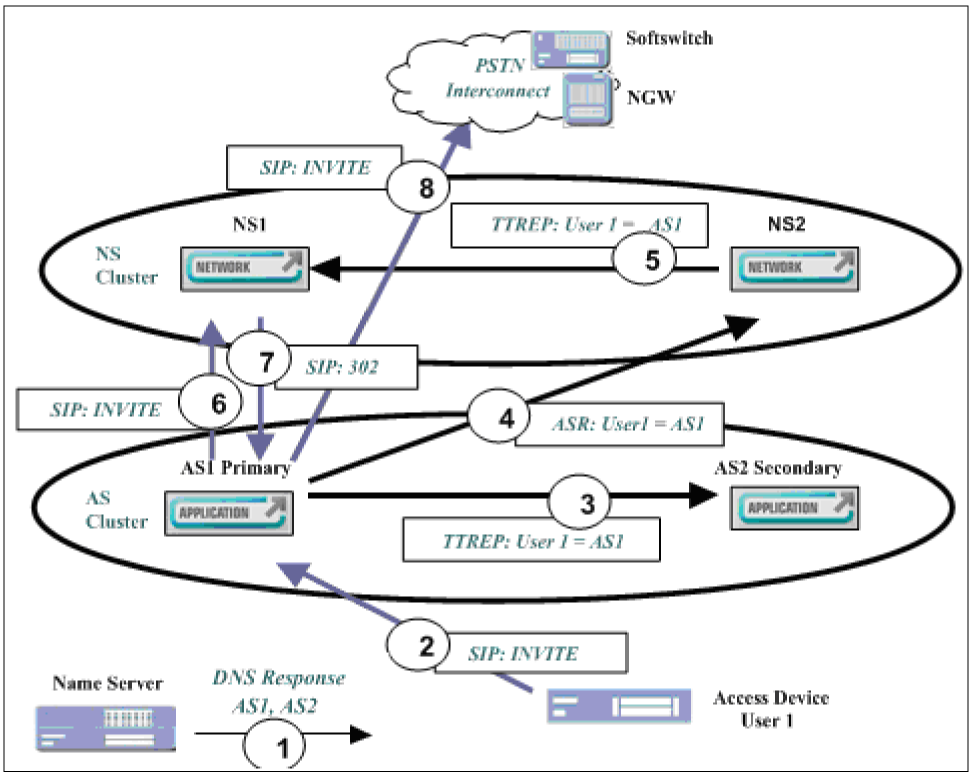

Figure 9 Application Server Recovery: User-Initiated Rollback

The sequence of events is as follows:

1. The access device performs a DNS lookup on the Application Server cluster FQDN. The name server returns AS1 followed by AS2 in the response.
2. The access device sends an INVITE to AS1. If geo-redundancy is enabled, then no SIP OPTIONS are sent. In such cases, rollback is automatic.
3. AS1 checks its database and discovers that User 1 is active on AS2. AS1 starts a rollback of the user, by updating its database to indicate that User 1 is now active on AS1 and informing AS2 via TTREP that User 1 has rolled back to AS1.

4. The primary Application Server AS1 informs the Network Server cluster of the user active node change, by sending an ASR message to one of the Network Servers in the cluster.
5. NS2 updates the database for User 1 to indicate that AS1 should again be the first contact. NS2 informs NS1 via TTREP that User 1 is now active on AS1. All calls destined to User 1 cause the Network Server cluster to return a contact with AS1 followed by AS2.
6. AS1 sends an INVITE to the Network Server cluster.
7. NS2 returns a 302 TEMPORARILY MOVED with contact equal to the NGW.
8. AS1 sends an INVITE to the PSTN NGW.

4.1.32 APPLICATION SERVER GEO-REDUNDANCY PROXY: CALL ORIGINATION

The scenario for this failure is that User 1 initiates a call to a PSTN subscriber and the primary Application Server is unreachable from the access device but reachable from the secondary Application Server. In this scenario, the Geo-Redundancy Proxy is enabled.

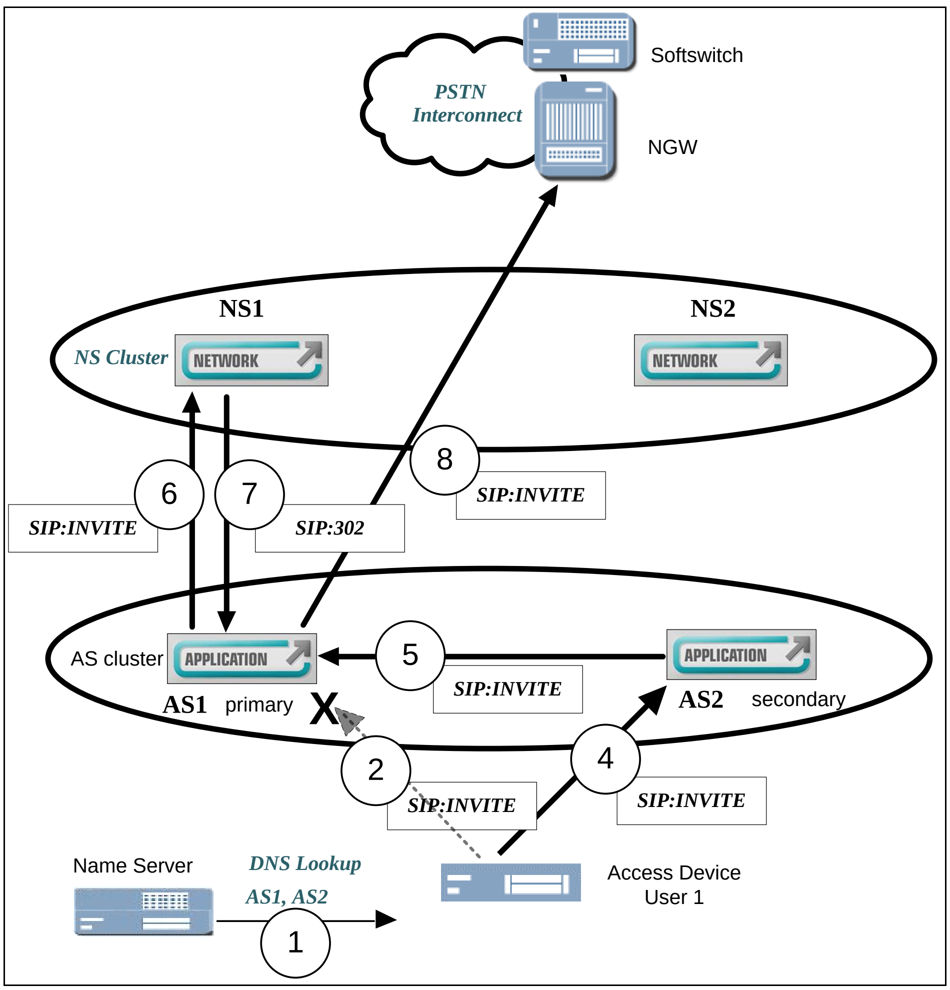

Figure 10 Application Server Proxy: Call Origination

The sequence of events is as follows:

1. The access device performs a DNS lookup on the Application Server cluster FQDN. The name server returns AS1 followed by AS2 in the response.
2. The access device sends an INVITE to AS1. AS1 is in service but unreachable due to network conditions.
3. The access device times out on AS1 and route advances to AS2. The access device sends an INVITE to AS2.
4. AS2 receives the INVITE and determines to relay the request to AS1 since AS1 is available. AS2 does not update its database to indicate that User 1 is active on AS2.
5. AS1 receives the INVITE and sets the isReachableFromPrimary indicator to “false”.
6. AS1 sends an INVITE to the Network Server cluster.
7. NS2 returns a 302 TEMPORARILY MOVED with a contact equal to the NGW.
8. AS1 sends an INVITE to the PSTN NGW.

4.1.33 APPLICATION SERVER PROXY: CALL TERMINATION

The scenario for this failure is that a PSTN subscriber initiates a call to User 1. User 1 is not reachable from the primary Application Server but is reachable from the secondary Application Server. User 1’s isReachableFromPrimary indicator is set to “false”. In this scenario, the geo-redundancy proxy is enabled.

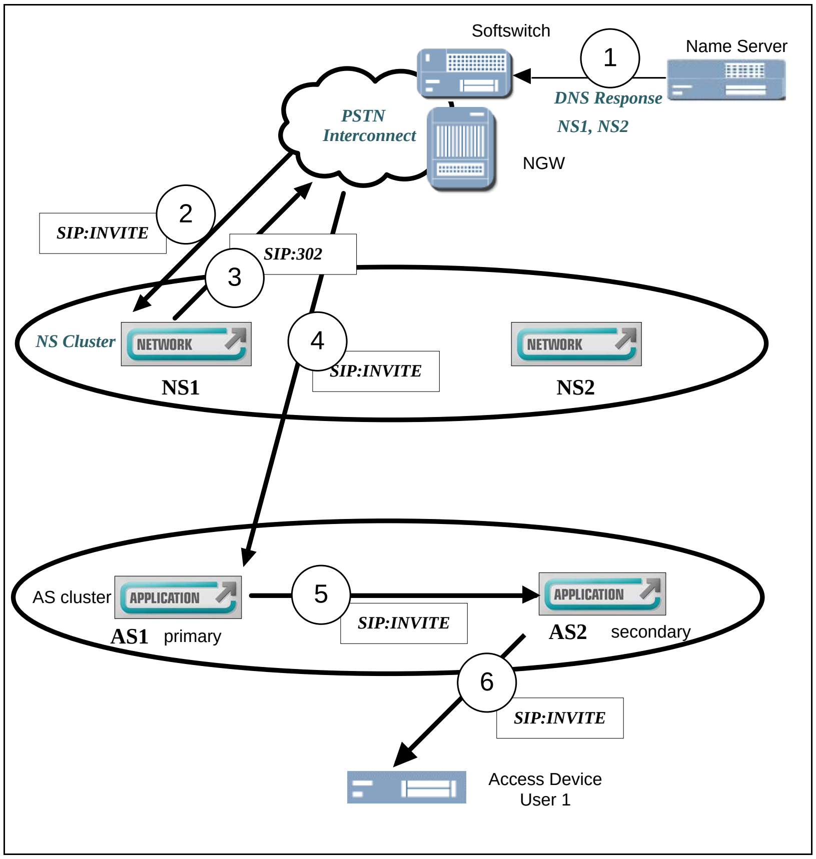

Figure 11 Application Server Proxy: Call Termination

The sequence of events is as follows:

1. The network device performs a DNS lookup on the Network Server cluster FQDN. The name server returns NS1 followed by NS2 in the response.
2. The network device sends an INVITE destined for User 1 to NS1.
3. The network device returns a 302 TEMPORARILY MOVED message, indicating that User 1 is located on AS1 followed by AS2.
4. The network device sends an INVITE to AS1. The network device sends an INVITE to AS2.
5. AS1 receives the INVITE and issues an INVITE for User 1. Since User 1's isReachableFromPrimary indicator is set to "false", and since AS2 is available, AS1 sends the INVITE to AS2 proxy.
6. AS2 relays the INVITE to User 1's access device.

5 DNS REQUIREMENTS

The Clearspan redundancy solution relies on DNS for proper operations. This section outlines the redundancy DNS requirements.

DNS A AND SRV RESOURCE RECORDS

The A Resource Record (RR) is used to map a name to an address. For Clearspan, A records are used to provide mappings for the individual server names to IP addresses, and to provide an A record that shows a mapping for all members of the cluster. An A record used by Clearspan would look as follows:

```
PrimaryAS.company.com.          IN      A       10.1.1.1
SecondaryAS.company.com.        IN      A       10.1.2.1
as.company.com                  IN      A        10.1.1.1
as.company.com                  IN      A        10.1.2.1
```

The SRV RR specifies the location of a server(s) for a specific service, protocol, and domain. For Clearspan, the service is SIP and protocol is UDP. SRV RR supports cost and weight values, allowing for the prioritization of contacts returned. An SRV record for Clearspan use would look as follows:

```
sip.udp.as.company.com. IN   SRV 1   50   5060   PrimaryAS.company.com.
sip.udp.as.company.com. IN   SRV 2   50   5060 SecondaryAS.company.com.
```

REQUIRED RESOURCE RECORDS

The following table identifies the DNS record requirements for different components in the Clearspan redundancy solution:

COMPONENT	TYPE OF DNS ENTRY	DESCRIPTION
Web browser	Single “A” record resolving to Application Server cluster peers public access addresses. Single “A” record resolving to Network Server cluster peers public access addresses. Individual resource record for each server to support HTTP redirects.	For web access, end users should not point directly to a specific Application Server or Network Server. Instead, a cluster A record DNS entry should be used. All web browsers support this. In event of a failure, it ensures automatic rollover to an alternate server for web access. Application Server and Network Server cluster “A” records can be returned in round robin fashion.
Access devices	Single resource record for the Application Server cluster that resolves to the primary Application Server followed by the secondary Application Server.	Access devices use DNS to identify the Application Server cluster that should be used to process calls. A cluster resource record should be configured to ensure the primary Application Server address always has a higher q-value. This ensures:

COMPONENT	TYPE OF DNS ENTRY	DESCRIPTION
		• A forced primary/backup Application Server model. • Access device-initiated rollbacks from a backup to a primary Application Server. Depending on the access device DNS capabilities, the following cluster resource record may be required: "SRV" records that resolve to the Application Server is signaling addresses. The primary Application Server is returned with a lower cost. • Fixed "A" record that resolve to the Application Server's signaling addresses. The primary Application Server always returns first.
Network devices	Single resource record that resolves to the Network Server cluster.	Network devices use DNS to identify the Network Server cluster that should be used. Network Server cluster members can be returned in any order and can be returned in round-robin fashion to support load balancing. Depending on the network device DNS capabilities, the following cluster FQDN may be required: • "SRV" records that resolve to the Network Server is signaling addresses. • "A" record that resolves to the Network Server is signaling addresses.
Application Server and Network Server internal communication	Individual "A" resource record for each server.	Each A resource record must be able to uniquely resolve to a specific server. These individual server resource records are required for: • Call processing queries • Sync API database updates • Location API queries • ASR updates for end-user rollbacks
Failover CDR support	Reverse Application Server cluster FQDN that resolves to the secondary Application Server followed by the primary server.	Reverse Application Server cluster FQDN is sent in the contact field for SIP messages coming from the secondary Application Server. The Application Server cluster FQDN is sent in the contact field for SIP messages coming from the primary Application Server.
Conferencing Server	Single resource record that resolves to the conferencing server cluster.	The Application Servers use DNS to identify the Conferencing Server cluster that should be used. Conferencing Servers are deployed in load-balancing configuration for provisioning and in primary/secondary configuration for signaling. The following cluster FQDN may be required: • "SRV" record that resolves to the conferencing server is signaling

COMPONENT	TYPE OF DNS ENTRY	DESCRIPTION
		addresses. • "A" record that resolves to the conferencing server is provisioning addresses.
Element Management System server	Single resource record that resolves to the EMS server cluster.	The EMS server's clients may use a cluster FQDN to connect to the EMS. An "A" record may be used to resolve the EMS addresses.
Profile Server	Single resource record that resolves to the Profile Server cluster.	The Web Server uses DNS to identify the Profile Server cluster that should be used. Profile Server cluster "A" records can be returned in round robin fashion.

DEVICE RESOURCE RECORD REQUIREMENTS

Different devices support different DNS resource record lookups. SIP devices support both SRV and A record lookups (performing an SRV lookup before an A record lookup).

Given that the Clearspan redundancy solution requires that the primary Application Server is always the first server attempted, SRV records should be used whenever possible.

For devices that support only A record lookups, the Application Server cluster A resource records must be returned in fixed order. For more information on fixed A records, see section *Fixed A Records and BIND*.

Some devices do not support redundancy via DNS, but rather provide for the configuration of multiple call agent/proxy using IP addresses. In this case, the devices should be pointed to the respective cluster member IP addresses (the Network Server cluster for network devices and Application Server cluster for access devices).

Consult the appropriate *Clearspan Partner Configuration Guide* for detailed information on specific device redundancy requirements, capabilities, and setup.

5.1.1 DEVICE FAILOVER TUNING

For the redundancy solution to operate transparently from an end-user perspective, failovers from a primary to secondary Application Server must occur in a reasonable amount of time.

A device should failover to the backup/secondary Application Server within one to two seconds. This period is long enough to avoid unnecessary failovers due to signaling latency, yet short enough to not impact end-user perception.

Device failover tuning consists of adjusting the number of retries a device attempts before timing out on the primary Application Server and route advance to the secondary/backup Application

Server. For more information on failover tuning for a specific device, see the appropriate *Clearspan Partner Configuration Guide*.

FIXED A RECORDS AND BIND

Fixed A records mapping the Application Server cluster name to the primary and secondary servers are required for all access devices that are limited to A record lookup support.

In general, most existing DNS servers return multiple A records' results for the same name in a round-robin fashion. To return records in fixed order as they appear in the zone file, the `rrset-order fixed` command must be used. This can be applied to the entire zone or specific FQDNs. A sample of the BIND `named.conf rrset-order` command follows:

```
rrset-order {
class IN type A name "as.company.com" order fixed;
};
```

The *rrset-order fixed* is supported in BIND 8. BIND 9 does not support the *rrset-order fixed* command.

DNS DEPLOYMENT CONSIDERATIONS

How DNS for Clearspan is deployed depends on the network configuration and supported devices. Generally, there are two approaches to meeting Clearspan DNS requirements:

- Overlay the Clearspan zone on an existing DNS infrastructure.
- Create a new independent DNS infrastructure to support Clearspan DNS requirements.

In some cases, a combination of the two is the best solution.

5.1.2 OVERLAY CLEARSPAN ZONE ON EXISTING DNS INFRASTRUCTURE

A deployment that uses SIP devices supporting SRV records, and does not have fixed A records requirements can generally overlay Clearspan DNS requirements over an existing DNS infrastructure.

- The Clearspan zone is set up on an existing name server. The following resource records are set up:
  - Application Server cluster SRV record that returns primary Application Server with lower cost
  - Reverse Application Server cluster SRV record that returns secondary Application Server with lower cost
  - Network Server cluster SRV record that returns Network Server cluster peers in any order

  - A records that resolve to each Application Server and Network Server cluster peer member
  - Application Server cluster A records for web access. Records can be returned in any order.
  - Network Server cluster A records for web access (optional). Records can be returned in any order.

- Since no fixed A record is required, either BIND 8 or BIND 9 can be used.
- The Clearspan zone is registered with its parent domain server; therefore, zone information can be resolved from any local name server on the Internet. This allows SIP devices to be deployed in a private space and have any DNS server via DHCP, and still resolve the Application Server cluster SRV record in the proper order.

5.1.3 OVERLAY CLEARSPAN ZONE ON EXISTING DNS INFRASTRUCTURE

A deployment that uses SIP devices supporting SRV records, and does not have fixed A records requirements can generally overlay Clearspan DNS requirements over an existing DNS infrastructure.

- The Clearspan zone is set up on an existing name server. The following resource records are set up:
  - Application Server cluster SRV record that returns primary Application Server with lower cost
  - Reverse Application Server cluster SRV record that returns secondary Application Server with lower cost
  - Network Server cluster SRV record that returns Network Server cluster peers in any order
  - A records that resolve to each Application Server and Network Server cluster peer member
  - Application Server cluster A records for web access. Records can be returned in any order.
  - Network Server cluster A records for web access (optional). Records can be returned in any order.

- Since no fixed A record is required, either BIND 8 or BIND 9 can be used.
- The Clearspan zone is registered with its parent domain server; therefore, zone information can be resolved from any local name server on the Internet. This allows SIP devices to be deployed in a private space and have any DNS server via DHCP, and still resolve the Application Server cluster SRV record in the proper order.

This approach can also be used when the MGCP device supports static entry of multiple call agents (DNS is not required). For more information on call agent identification requirements for a specific device, see the appropriate *Clearspan Partner Configuration Guide*.

6 REDUNDANCY CONFIGURATION

This section outlines Clearspan redundancy configuration requirements. The following areas are covered:

- Linux configuration
- DNS configuration
- Network Server configuration
- Application Server configuration
- Profile Server configuration
- Cluster data replication requirements
- ID uniqueness requirements across Application Server clusters

A sample Clearspan network is provided and used in all configuration examples. The sample network consists of a redundant Network Server cluster, a redundant Application Server cluster, two Media Servers that are pooled, and a DNS server that supports all Clearspan resource record requirements.

The sample network also assumes a common signaling plane interface deployment. Each server has one public interface, with no multipathing.

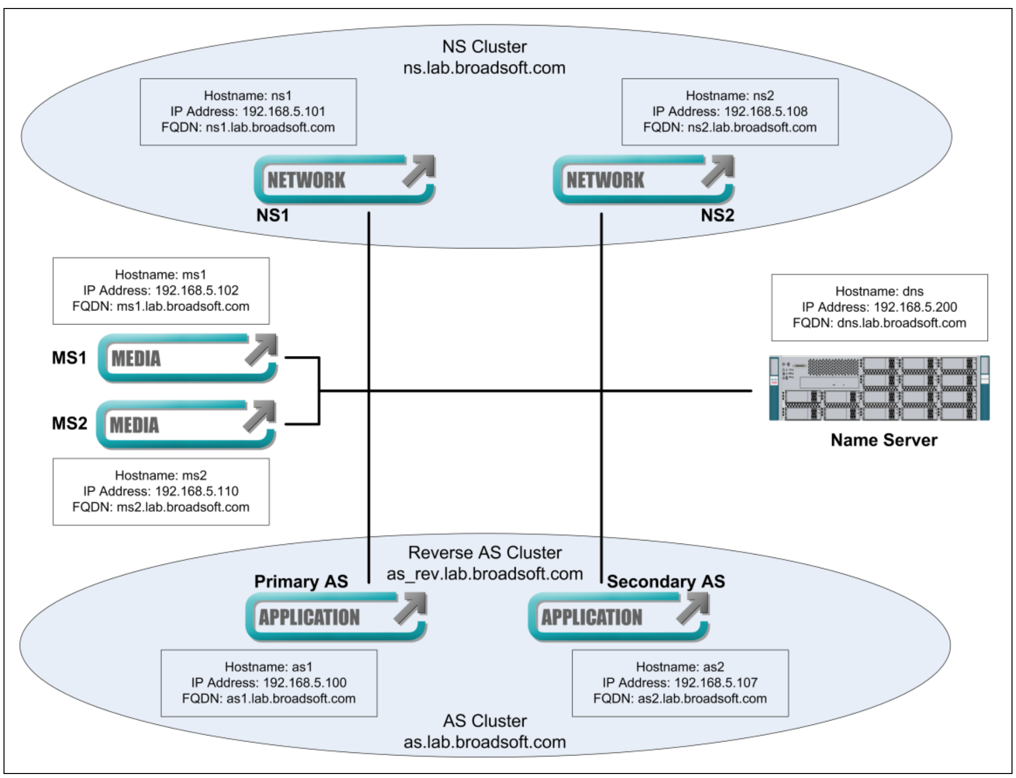

Figure 12 Redundancy Configuration

LINUX CONFIGURATION

This section outlines the platform configuration setup required to support redundancy. These configuration steps are performed prior to installing Clearspan and are explained here for information only.

6.1.1 FILE/ETC/HOSTS

The `/etc/hosts` file for each member in a cluster includes entries for all other cluster peer members. The entry representing the primary interface of the server should have an individual FQDN as its first entry and its short hostname as an alias. The Media Servers do not need to know about any of the other Media Servers and do not require FQDNs in the `/etc/hosts` file.

```
************* AS1 Hosts file **************
as1$ more hosts
# Internet host table
#
127.0.0.1       localhost
192.168.5.100   as1.lab.clearspan.com as1   192.168.5.107   as2

************* AS2 Hosts file **************
as2$ more hosts
# Internet host table
#
127.0.0.1       localhost
192.168.5.107   as2.lab.clearspan.com as2   192.168.5.100   as1

************* NS1 Hosts file **************
ns1$ more hosts
# Internet host table
#
127.0.0.1       localhost
192.168.5.101   ns1.lab.clearspan.com ns1   192.168.5.108   ns2

************* NS2 Hosts file **************
ns2$ more hosts
# Internet host table
#
127.0.0.1       localhost
192.168.5.108   ns2.lab.clearspan.com ns2   192.168.5.101   ns1

************* MS1 Hosts file **************
ms1$ more hosts
# Internet host table
#
127.0.0.1       localhost
192.168.5.102   ms1.lab.clearspan.com ms1
************* MS2 Hosts file **************
ms2$ more hosts
# Internet host table

127.0.0.1       localhost
192.168.5.110   ms2.lab.clearspan.com ms2
```

6.1.2 NETWORK TIMING PROTOCOL: TIME SYNCHRONIZATION BETWEEN SERVERS

For RSYNC to function properly, it is important that all servers have the same clock value. Enabling Network Timing Protocol (NTP) on all Clearspan servers accomplishes this. This also ensures that all logs and billing call detail records have the proper timestamp.

There are a number of ways to configure NTP. All are acceptable as long as server times are synchronized.

One Linux server acts as the NTP server synchronized to external time servers and the remaining servers act as clients.

The configuration for the clients and the server is very similar. The difference is that the server points to time reference servers and clients point to the NTP server.

The following lines show how you can configure NTP on Linux.

On the server:

1. Edit the */etc/ntp.conf* file and comment out the following lines:

```
#multicastclient
#broadcastclient
```

2. In the same file, add or modify the server lines to read:

```
# File /etc/ntp.conf
# Localhost timekeeper.foo.com
server tick.uh.edu
server time.nist.gov
server tick.usno.navy.mil
driftfile /etc/ntp.drift
```

The above example configuration defines three server sources for the “timekeeper” server.

On the clients:

1. Edit the */etc/ntp.conf* file and comment out the following lines:

```
#multicastclient
#broadcastclient
```

2. In the same file, add or modify the server line to read, server <ntp server>, where <ntp server> is the address of the NTP server. For example:

```
# File /etc/ntp.conf
# Localhost yogi.foo.com
server timekeeper.foo.com
server time.nist.gov
driftfile /etc/ntp.drift
```

In the above client configuration example, the clients point to the “timekeeper”, yogi.foo.com, as well as to a reference server (as a backup).

1. On all servers, configure NTP to start at boot time using the following command:
   `chkconfig --level 345 ntpd on`

2. Finally, restart the NTP daemon:
   `/etc/init.d/ntpd restart`

6.1.3 DNS LOOKUP CONFIGURATION

DNS lookup should be enabled on each Clearspan server using the following steps.

The `resolv.conf file` must be created in the `/etc` directory. This file identifies the name servers to be used for DNS lookups.

```
as1$ more resolv.conf
#
domain lab.company.com
nameserver      192.168.5.200
```

The domain name resolution mode must be configured to look in the local `/etc/hosts` first, and in the DNS server second. This is done by setting the `hosts` entry in the `nsswitch.conf` to “`hosts: files dns`”.

```
as1$ more nsswitch.conf
#
# /etc/nsswitch.files:
# An example file that could be copied over to /etc/nsswitch.conf; it
# does not use any naming service.
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet" transports.
passwd:     files
group:      files
hosts:      files dns
ipnodes:    files
networks:   files
protocols:  files
rpc:        files
ethers:     files
netmasks:   files
```

DNS CONFIGURATION

The DNS name server is an essential component in the Clearspan redundancy solution. This section outlines the configuration for the DNS name server. The configuration steps are as follows:

1. Create /etc/named.conf file.
2. Create the data files used by the name server.
3. Manage the name server.

6.1.4 FILE /ETC/NAMED.CONF

The `/etc/named.conf` file establishes the server as a primary, secondary, or cache-only name server. It also specifies the zones over which the server has authority and which data files it should read to get its initial data. This file is read by name server daemon (`in.named`) when it is started. In this file, it directs the `in.named` daemon to local data files for specified domain located in `/var/named` directory.

The `rrset-order {}` portion is where you specify that A records for zone “lab.company.com” should be returned in a fixed order instead of round-robin.

The following shows an example of the `/etc/named.conf` file used by the DNS server 192.168.5.200 that is used in the sample network server.

```
dns$  more /etc/named.conf
options {
        directory "/var/named";
        /*
         * If there is a firewall between you and nameservers you want
         * to talk to, you might need to uncomment the query-source
         * directive below.  Previous versions of BIND always asked
         * questions using port 53, but BIND 8.1 uses an unprivileged
         * port by default.
         */
        // query-source address * port 53;
        rrset-order {
                class IN type A name "lab.company.com" order fixed;
        };
};
//
// a caching only nameserver config
//
zone "." {
        type hint;
        file "db.cache";
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
};

zone "5.168.192.in-addr.arpa" {
        type master;
        file "db.192.168.5";
};

zone "lab.company.com" {
        type master;
        file "db.lab.company.com";
};
```

6.1.5 NAME SERVER DATA

There are four types of data files that a name server must have, as follows:

- **Root Server Files:** This file specifies name-to-address mapping of root servers. The information in this file is described as “hints” to the name daemon process as the name daemon process attempts to contact the servers listed, until one of them responds. The name daemon uses the list returned from the root server and not the servers specified in this file until the time-to-live expires.

  In the above example, the actual name of this file is db.cache. To create this file, you can either FTP the root server file from any working DNS server or go to ftp://ftp.rs.internic.net/domain/named.root.

```
dns$ more /var/named/db.cache
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;              submenu          InterNIC Registration Archives
;           file                named.root
;       last update:    Aug 22, 1997
;       related version of root zone:   1997082200
;
; formerly NS.INTERNIC.NET
;
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                        3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                        3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.      3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                        3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.      3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
;
.                        3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.      3600000      A     192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.                        3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.      3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                        3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.      3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                        3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.      3600000      A     192.36.148.17
;
; temporarily housed at NSI (InterNIC)
;
```

```
.                          3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.        3600000      A     198.41.0.10
;
; housed in LINX, operated by RIPE NCC
;
.                          3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.        3600000      A     193.0.14.129
;
; temporarily housed at ISI (IANA)
;
.                          3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.        3600000      A     198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                          3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.        3600000      A     202.12.27.33
; End of File
```

- **Domain-Info File:** This file contains the mappings of names to IP addresses for all systems in the domain being served by this name server (A records and SRV records). In addition, this file must specify an SOA record and Network Server record for all name servers for this domain. In our example, a zone file for "lab.company.com" is required.

```
dns$ more /var/named/db.lab.company.com
lab.company.com. IN SOA dns.lab.company.com. root.lab.company.com. (
                        2002041704 ; serial
                        900 ; refresh
                        3600 ; retry
                        604800 ; expire
                        86400 ; default_ttl
                        )
                                IN      NS      dns.lab.company.com.
as1.lab.company.com.          IN      A       192.168.5.100
as2.lab.company.com.          IN      A       192.168.5.107
ns1.lab.company.com.          IN      A       192.168.5.101
ns2.lab.company.com.          IN      A       192.168.5.108
cs1.lab.company.com.          IN      A       192.168.5.113
cs2.lab.company.com.          IN      A       192.168.5.114
ms1.lab.company.com.          IN      A       192.168.5.102
ms2.lab.company.com.          IN      A       192.168.5.110
dns.lab.company.com.          IN      A       192.168.5.200
as.lab.company.com.           IN      A       192.168.5.100
as.lab.company.com.           IN      A       192.168.5.107
ns.lab.company.com.           IN      A       192.168.5.101
ns.lab.company.com.           IN      A       192.168.5.108
cs.lab.company.com.           IN      A       192.168.5.113
cs.lab.company.com.           IN      A       192.168.5.114
_sip._udp.as.lab.company.com. IN SRV 1 50 5060  as1.lab.company.com.
_sip._udp.as.lab.company.com. IN SRV 2 50 5060  as2.lab.company.com.
_sip._udp.as_rev.lab.company.com. IN SRV 1 50 5060  as2.lab.company.com.
_sip._udp.as_rev.lab.company.com. IN SRV 2 50 5060  as1.lab.company.com.
_sip._udp.ns.lab.company.com. IN SRV 1 50 5060  ns1.lab.company.com.
_sip._udp.ns.lab.company.com. IN SRV 2 50 5060  ns2.lab.company.com.
_sip._udp.cs.lab.company.com. IN SRV 2 50 5060  cs1.lab.company.com.
_sip._udp.cs.lab.company.com. IN SRV 2 50 5060  cs2.lab.company.com.
```

- **Inverse-Domain-Info File:** This file contains mappings for address to name translation (PTR, pointer records). In this example, a file named `db.192.168.5` provides the pointer records.

```
dns$.more /var/named/db.192.168.5
5.168.192.in-addr.arpa. IN SOA dns.lab.company.com. root.lab.company.com.
(
                        2002041704 ; serial
                        900 ; refresh
                        3600 ; retry
                        604800 ; expire
                        86400 ; default_ttl
                        )
                IN      NS      dns.lab.company.com.

100             IN      PTR     as1.lab.company.com.
107             IN      PTR     as2.lab.company.com.
101             IN      PTR     ns1.lab.company.com.
108             IN      PTR     ns2.lab.company.com.
113             IN      PTR     cs1.lab.company.com.
114             IN      PTR     cs2.lab.company.com.
102             IN      PTR     ms1.lab.company.com.
110             IN      PTR     ms2.lab.company.com.
200             IN      PTR     dns.lab.company.com.
```

- **Loopback-Domain-Info Files**: This file is used to specify the inverse loopback domain address to name translation.

  In this example, the actual name of this file is db.127.0.0.

```
dns$  more /etc/named/db.127.0.0
0.0.127.in-addr.arpa. IN SOA dns.mtl.company.com. root.ihs.company.com (
                        2001011802 ; serial
                        10800 ; refresh
                        3600 ; retry
                        604800 ; expire
                        86400 ; default_ttl
                        )
0.0.127.in-addr.arpa.           IN      NS      dns.mtl.company.com.
1.0.0.127.in-addr.arpa.         IN      PTR     localhost.
```

6.1.6 MANAGE NAME SERVER

The BIND 8 name server is controlled using the name daemon control (`ndc`) utility.

```
   ndc  start           #  Start the name server
   ndc  stop            #  Stop the name  server
   ndc  restart         #  Bounce the name server. Required
                          #  whenever a file is changed.
```

Name server responses can be verified using the `nslookup` utility. The `nslookup` utility sends queries to the configured domain name server and can be run on any server.

NETWORK SERVER CONFIGURATION

This section outlines the Network Server configuration steps for Clearspan redundancy.

6.1.7 DEFINE NETWORK SERVER CLUSTER ALIAS

The Network Server allows an alias name to be added that can be used by routing network elements or hosting network elements to identify the Network Server cluster in incoming INVITEs. The alias table is automatically populated during installation with the peer server's IP address and FQDN. The Network Server cluster FQDN (ns.lab.company.com) must be manually added.

```
NS_CLI/System/Alias> get

127.0.0.1
192.168.5.101
192.168.5.108
ns1.lab.company.com
ns2.lab.company.com
ns.lab.company.com
```

6.1.8 DEFINE CLUSTER OF APPLICATION SERVERS

Application Servers are configured as hosting network elements on the Network Server. A redundant Application Server cluster is made of two peers, which are mapped to one hosting network element with two nodes (node 0 for the primary Application Server and node 1 for the secondary Application Server) on the Network Server. Each node has its own set of addresses allowing the Network Server to route calls to a specific server in the Application Server cluster.

```
NS_CLI/System/Device/HostingNE>  get hostNE ASCluster

Hosting Network Element  ASCluster
   Type                        =  named
   Profile                     =  Hosting
   Default Enterprise          =  NIL_ENTERPRISE
   Default Routing Enterprise =  NIL_ENTERPRISE
   Default Site                =  DFLT_SITE
   Default Routing Site        =  DFLT_SITE
   Poll                        =  false
   OpState                     =  enabled
   State                       =  OnLine
   Signaling Attributes        =  CallTypeInfoRequired
   Country Code                =  1
   XSP Version Equal           =  false
   Cluster Type                =  primarySecondary
   User Capacity (thousands)  =  10000
   Hosting NE Capabilities     =  ProvisioningCapable,
CallProcessingCapable

NS_CLI/System/Device/HostingNE/Node> get

  HostingNe  NodeID  default                   description
==========================================================
  ASCluster      0     true     Node 0 (automatically added)
  ASCluster      1     false

NS_CLI/System/Device/HostingNE/Address> get

  HostingNe  NodeID           Address         type   cost  weight  port
=========================================================================
=
 ASCluster    0         192.168.5.100     Signaling     1      50  5060
 ASCluster    0                   as1         Alias     -       -     -
 ASCluster    0   as1.lab.company.com       Access     1      50  5060
 ASCluster    1         192.168.5.107     Signaling     1      50  5060
 ASCluster    1                   as2         Alias     -       -     -
 ASCluster    1   as1.lab.company.com       Access     1      50  5060
```

It is important to note that:

- The primary Application Server must be the default node, that is, node 0.
- The hosting network element, *ASCluster,* must have the session replication disabled (the *sessionReplicationEnabled* parameter set to “false”).
- The address types have the following significance:

TYPE	DESCRIPTION
Signaling	This address is a SIP signaling address. It can be entered in the contact list of a returned 302 following a SIP request.
Access	This address is a public address (that is, Web Server address). It is returned when requesting a Hosting NE address through the Location API. It is not returned for SIP requests.
DualRouting	This address is public and is used for signaling at the same time. It is returned for both SIP and location API requests.
Alias	This alias can be used to identify a hosting network element. The Network Server never returns this in the contact list. It can be used for: Sync API – The canonical hostname of the Application Server synchronized with the Network Server must have an alias defined (for example, AS1 and AS2 for ASCluster). Incoming call requests – An address that is recognized as valid for sending requests to the Network Server.

6.1.9 DEFINE DEFAULT ENTERPRISE

To support proper redundancy failover, all stand-alone groups must belong to a default enterprise that has the ExtDialing private policy assigned. This ensures that intra-group calls between endpoints hosted on different Application Server cluster members during an endpoint failover work. In addition, any other enterprises that are created for the sharing of private policies must have ExtDialing assigned.

The ExtDialing private policy is used to route intra-group calls between two Application Servers in a cluster when an end user in a group is migrated to the backup Application Server, while other users are still supported on the primary Application Server. This policy also provides support for intra-group calls between phantom users belonging to the same enterprise.

Once the default enterprise is created and has ExtDialing assigned, the Application Server hosting network element's `Default Enterprise` parameter should be set to the default enterprise name. This ensures that all new groups created on the Application Server are automatically added to the default enterprise.

```
NS_CLI/SubscriberMgmt/Enterprise> get
Enterprise  Default_Ent
   Call Logging         =  Enabled
   Description          =  Default Enterprise
   Intra-LATA PIC       =  NILCAC
   Inter-LATA PIC       =  NILCAC
   International PIC    =  NILCAC
   Routing profile      =  Profall
   LCA ID               =
   Source ID            =
   SourceidForEAPrefix =
   Private policies     =  {voiceVPN, ExtDialing}
   KeepGroupsTogether   =  false
   Hosting NE Set       =
   Current Hosting NE Node  =  'NOT ASSIGNED'

NS_CLI/System/Device/HostingNE> get
Retrieving data... Please wait...

Hosting Network Element  ASCluster
   Type                         =  named
   Profile                      =  Hosting
   Default Enterprise           =  NIL_ENTERPRISE
   Default Routing Enterprise =  NIL_ENTERPRISE
   Default Site                 =  DFLT_SITE
   Default Routing Site         =  DFLT_SITE
   Poll                         =  false
   OpState                      =  enabled
   State                        =  OnLine
   Signaling Attributes         =  CallTypeInfoRequired
   Country Code                 =  1
   XSP Version Equal            =  false
   Cluster Type                 =  primarySecondary
   User Capacity (thousands)  =  10000
   Hosting NE Capabilities     =  ProvisioningCapable,
CallProcessingCapable
```

APPLICATION SERVER CONFIGURATION

This section outlines the Application Server configuration steps for Clearspan redundancy.

6.1.10 DEFINE APPLICATION SERVER CLUSTER ALIAS

The Application Server allows an alias name to be added that can be used by access devices to identify the Application Server cluster in incoming call requests. The alias table is automatically populated during the installation with the peer server’s IP address and the default domain. The Application Server cluster FQDN (*as.lab.company.com*) must be manually added.

```
AS_CLI/System/Alias> get

127.0.0.1
192.168.5.100
192.168.5.107
lab.company.com
as.lab.company.com
```

6.1.11 ENABLE SRV FOR APPLICATION SERVER

The Application Server supports SRV lookups for SIP signaling. This can be enabled by setting the `supportDnsSrv` parameter under `AS_CLI/Interface/SIP>` to “true”.

```
AS_CLI/Interface/SIP> get
  t1 = 500
  t2 = 4000
  maxRegistrationTime = 86400
  maxForwardingHops = 10
  inviteAuthenticationRatio = 0
  encryptFromHeader = false
  sessionExpiresMinimum = 60
  sessionExpiresTimer = 900
  sessionAuditAllowed = true
  accessControl = false
  sendE164 = false
  suspiciousAddressThreshold = 0
  privacyVersion = privacy-03
  privacyEnforceScreening = false
  listeningPort = 5060
  networkProxyHost =
  networkProxyPort =
  accessProxyHost =
  accessProxyPort =
  supportDnsSrv = true
  maxAddressesPerHostname = 10
  maxAddressesPerHostnameInDialog = 4
  useDomainForRealm = true
  defaultRealm = Named
  includeT38CapabilityInfo = false
```

6.1.12 SET GEO-REDUNDANCY PARAMETERS

The proper functioning of geo-redundancy required that the following Application Server parameters be set as specified as follows:

- The *enabled* parameter under *AS_CLI/System/Session data replication*> must be set to “false”.

```
AS_CLI/System/SessionDataReplication> get
  enabled = false
  connectionPort = 5015
```

- The *clusterAddress* parameter under *AS_CLI/Interface/SIP*> must be empty.

```
AS_CLI/Interface/SIP> get
  t1 = 2000
  t2 = 4000
  maxForwardingHops = 5
  inviteAuthenticationRatio = 1.0
  encryptFromHeader = false
  100rel = true
  useDomainForSubscriberAddress = false
  accessControl = false
  sendE164 = false
  suspiciousAddressThreshold = 0
  privacyVersion = RFC3323
  privacyEnforceScreening = false
  listeningPort = 5060
  networkProxyHost =
  networkProxyPort = 5060
  networkProxyTransport = unspecified
  accessProxyHost =
  accessProxyPort = 5060
  accessProxyTransport = unspecified
  supportDnsSrv = true
  ...
  send181Response = false
  routeToTrunkingDomainByDefault = false
  clusterAddress =
```

6.1.13 DEFINE NETWORK SERVER CLUSTER

The Application Server sends all inter-group and PSTN-bound calls to the Network Server. To avoid querying the DNS each time, you must define each Network Server by its IP address on the Application Server. Use the following CLI contexts:

- `AS_CLI/System/Device/NetServ/Routing>`

  Each time the Application Server needs to send a call to the Network Server, it uses a round-robin algorithm to select a Network Server from the list provisioned at this level.

- `AS_CLI/System/Device/NetServ/Sync>`

  The Application Server uses the list provisioned at this level to select a Network Server that supports SYNC API events. It is recommended that you define a preferred Network Server for synchronization traffic. The Application Server favors the preferred server, falling back to the next Network Server in the list when the preferred server is unavailable. Every 60 seconds thereafter, the Application Server attempts to re-establish contact with the preferred synchronization server over the BCCT synchronization channel.

The routing table under `AS_CLI/System/CallP/Routing/RoutingXLA>` must be updated to send all non-intra-group traffic to the Network Server.

```
AS_CLI/System/Device/NetServ/Routing> get
Net Address Port Transport Poll OpState Description
============================================================
192.168.5.101 UDP true enabled
192.168.5.108 UDP true enabled

AS_CLI/System/Device/NetServ/Synch> get

Preferred Update Network Server = 192.168.5.101

Net Address Port Description
=================================
192.168.5.101
192.168.5.108

AS_CLI/System/CallP/Routing/RoutingXLA> get

  NPA-NXX           Route
  =======================
     *              Network Server
```

6.1.14 AUTOMATIC ROLLBACK

For normal operation, the cluster's primary Application Server processes all calls. When the primary Application Server is unavailable, end users have their calls processed by the secondary Application Server (a rollover condition).

The secondary Application Server has an automated mechanism to migrate users back from the secondary to the primary Application Server when the primary Application Server becomes available again.

The automatic user rollback occurs if:

- The primary Application Server is available.
- An ASR rollback timer has expired.
- A user is currently active on the secondary server,

  and

- The user is not active on a call.

The automatic rollback mechanism is controlled through a configurable timer. By default, it runs every fifteen minutes. An operator can change the rollback timer using the Application Server command line interface.

```
AS_CLI/System/Redundancy> get
  redunRollBackTimer = 15

AS_CLI/System/Redundancy> set redunRollBackTimer 20
...Done

AS_CLI/System/Redundancy> get
  redunRollBackTimer = 20
```

Note that this same timer is also used by the geo-redundancy proxy for testing device reachability as described in section 3.7.2 Call Termination.

6.1.15 MGCP CONFIGURATION

The Clearspan MGCP interface must be configured to enable redundancy with the devices. Under the *AS_CLI/Interface/MGCP>* level, the *sendNotifiednotifiedEntity* parameter must be set to “true” and the *notifiedEntity* parameter must be set to the Application Server cluster FQDN. The FQDN must be provisioned in the DNS and have a DNS entry with an A record, which resolves to both the primary and secondary Application Servers in the redundant cluster.

```
AS_CLI/Interface/MGCP> get

  t1 = 500
  t2 = 4000
  listeningPort = 2727
  accessControl = false
  sendNotifiedEntity = true
  notifiedEntity = as.lab.clearspan.com
  suspiciousAddressThreshold = 1
  maxAddressesPerHostname = 10
  proxyAddress =
  embedDigitCollection = false
```

The *notifiedEntity* parameter can be used by the call agent to update the controlling call agent or redirect an endpoint to another call agent. In the Clearspan case, it can only be set to one value across the two Application Servers since it is replicated in the database and its specific purpose is for devices that do not allow both call agents to be manually configured. It requires the call agent to pass an FQDN that would resolve to a list of call agents (AS1 and AS2). This device would consider the call agent that sent the *notifiedEntity* as the controlling agent as long as it was one of the resolved notifiedEntity agents.

If a device allows for the manual configuring of both agents, then it enables the *notifiedEntity* since the endpoint already has the full list of call agents, and since by default, if *notifiedEntity* is not present, an MGCP endpoint considers the source of the request (for example, RQNT, CRCX) as the controlling agent. Therefore, if AS2 contacts the endpoint in the event of an AS1 failure, it automatically becomes the new controlling agent.

6.1.16 SIP CONFIGURATION

The Application Server populates the FROM and TO host portion of SIP messages with the public IP address of the Application Server. The public address is defined under

`AS_CLI/System/StartupParam>` parameter `publicIPAddress`. This parameter is also used in the VIA header. By default, the `publicIPAddress` parameter is set to the IP address of the public interface and it should not be changed. For more information on required Application Server public IP address settings, see the appropriate *Clearspan Partner Configuration Guide*.

The `publicIPAddress` parameter is not replicated across servers. This change, if required, must be done on each Application Server peer, and requires a reset of Clearspan to activate.

```
************ Primary Application Server -> as1 *********
AS_CLI/System/StartupParam> get
publicIPAddress = 192.168.5.100

************ Secondary Application Server -> as2 *********
AS_CLI/System/StartupParam> get
publicIPAddress = 192.168.5.107
```

The Application Server populates the CONTACT host portion of SIP messages with the `bw.sip.accessclustercontacthost` parameter value. The CONTACT header is what the device uses as the destination for all subsequent requests related to the same dialog (for example, BYE messages). By default, the contact host is equal to the public IP address of the peer. The `bw.sip.accessclustercontacthost` parameter should be set to the Application Server cluster FQDN on the primary Application Server, and the reverse Application Server cluster FQDN on the secondary Application Server (resolves to secondary followed by primary servers). This is required for the failover CDR generation to function properly in either direction (primary-to-secondary failover, secondary-to-primary failover).

Along with setting the access contact, the access contact port also needs to be set. The `bw.sip.accessclustercontactport` parameter needs to be cleared to ensure the access device does a SRV record lookup on the contact host.

The host contact parameters are unique to each Application Server peer. These changes must be done on each Application Server peer and requires a reset of Clearspan to activate.

The `bw.sip.networkclustercontacthost` and `bw.sip.networkclustercontactport` startup parameters perform the same function on the network side to a softswitch, network gateway, or Network Server as the `bw.sip.accessclustercontacthost` does on the access side, which sets the contact header to be the cluster contact. For example, for an Application Server with only a public/primary interface, set the `bw.sip.networkcluster contacthost` to the same value as the `bw.sip.accessclustercontacthost` (forward FQDN on the primary and reverse FQDN on the secondary). For a system with both a public/primary and private/primary interface, set the `bw.sip.networkcluster contacthost` to the FQDN used in the private network.

```
************ Primary Application Server -> as1 *********
AS_CLI/System/StartupParam>set   bw.sip.accessclustercontacthost
as.lab.company.com
AS_CLI/System/StartupParam>clear bw.sip.accessclustercontactport
AS_CLI/System/StartupParam> get
  java.ldap.connect.timeout = 5
  mail.pop3.timeout = 15000
  bw.sip.networkclustercontacthost =
  RemoteGroupPrefix = false
  publicIPAddress = 192.168.5.100
  ps.debug.config.file = /usr/local/named/bw_base/conf/PSDebugConfig.xml
  Accounting.active = true
```

```
  sun.net.inetaddr.ttl = 600
  bw.sip.accessclustercontactport =
  bw.callhalf.numThreads = 2
  mail.pop3.connectiontimeout = 15000
  %NOMIGRATE = <your property here!>
  xs.debug.config.file = /usr/local/named/bw_base/conf/XSDebugConfig.xml
  bw.sip.numThreads = 1
  bw.mgcp.numThreads = 1
  bw.sip.accessclustercontacthost = as.lab.company.com
  java.naming.factory.initial = com.sun.jndi.cosnaming.CNCtxFactory
  bw.database = AppServer
  mail.imap.timeout = 15000
  dtdLocation = /usr/local/named/bw_base/conf/cpl.dtd
  java.naming.provider.url = iiop://localhost:1050
  mail.imap.connectiontimeout = 15000
  bw.sip.networkclustercontactport =

************ Secondary Application Server -> as2 *********
AS_CLI/System/StartupParam>set   bw.sip.accessclustercontacthost
as_rev.lab.company.com
AS_CLI/System/StartupParam>clear bw.sip.accessclustercontactport
AS_CLI/System/StartupParam> get
  java.ldap.connect.timeout = 5
  mail.pop3.timeout = 15000
  bw.sip.networkclustercontacthost =
  RemoteGroupPrefix = false
  publicIPAddress = 192.168.5.100
  ps.debug.config.file = /usr/local/named/bw_base/conf/PSDebugConfig.xml
  Accounting.active = true
  sun.net.inetaddr.ttl = 600
  bw.sip.accessclustercontactport =
  bw.callhalf.numThreads = 2
  mail.pop3.connectiontimeout = 15000
  %NOMIGRATE = <your property here!>
  xs.debug.config.file = /usr/local/named/bw_base/conf/XSDebugConfig.xml
  bw.sip.numThreads = 1
  bw.mgcp.numThreads = 1
  bw.sip.accessclustercontacthost = as_rev.lab.company.com
  java.naming.factory.initial = com.sun.jndi.cosnaming.CNCtxFactory
  bw.database = AppServer
  mail.imap.timeout = 15000
  dtdLocation = /usr/local/named/bw_base/conf/cpl.dtd
  java.naming.provider.url = iiop://localhost:1050
  mail.imap.connectiontimeout = 15000
  bw.sip.networkclustercontactport =
```

6.1.17 USE USER DOMAIN NAME IN SIP REALM

The Application Server supports sending the user's domain name as realm information, as part of the SIP registration challenge. This can be enabled by setting the `useDomainForRealm` parameter under `AS_CLI/Interface/SIP>` to "true".

```
AS_CLI/Interface/SIP> get
  t1 = 500
  t2 = 4000
  maxRegistrationTime = 86400
  maxForwardingHops = 10
  inviteAuthenticationRatio = 0
  encryptFromHeader = false
  sessionExpiresMinimum = 60
```

```
sessionExpiresTimer = 900
sessionAuditAllowed = true
accessControl = false
sendE164 = false
suspiciousAddressThreshold = 0
privacyVersion = privacy-03
privacyEnforceScreening = false
listeningPort = 5060
networkProxyHost =
networkProxyPort =
accessProxyHost =
accessProxyPort =
supportDnsSrv = true
maxAddressesPerHostname = 10
maxAddressesPerHostnameInDialog = 4
useDomainForRealm = true
defaultRealm = Named
includeT38CapabilityInfo = false
```

6.1.18 GEO-REDUNDANCY PROXY CONFIGURATION

This section outlines the steps required to enable the geo-redundancy proxy functionality.

6.1.18.1 Enable geoProxy

The proxy functionality is enabled by setting the enabled parameter under `AS_CLI/System/Redundancy/GeoProxy>` to “true”.

```
AS_CLI/System/Redundancy/GeoProxy> get

  enabled = true
```

6.1.18.2 Identity Peer SIP Network Interface

Each Application Server must have information about its peer SIP network interface. This is done by configuring the `bw.sip.peernetworkinterfacehost` parameter under `AS_CLI/System/StartupParam`. This change must be done on each Application Server peer.

```
************ Primary Application Server -> as1 *********

AS_CLI/System/StartupParam> get

bw.sip.peernetworkinterfacehost = 192.168.5.107

************ Secondary Application Server -> as2 *********

AS_CLI/System/StartupParam> get

bw.sip.peernetworkinterfacehost = 192.168.5.100
```

6.1.18.3 Specify Record-Route Parameters

When relaying a message from or to the primary Application Server, the secondary Application Server uses the Record-Route mechanism to remain in the signalling path for the duration of the SIP dialog. The Record-Route header is populated by adding a new Record-Route entry to a request, or rewriting the Record-Route entry in a response. The server sets the entry's URI based on the setting of a collection of startup parameters described in the following table.

PARAMETER	DESCRIPTION AND ALTERNATIVES
bw.sip.accessrecordroutehost	This parameter has the same meaning as the bw.sip.accessclustercontacthost; however, it is used by the secondary Application Server proxy to populate a Record-Route entry's host. Populating with the reverse cluster access-side hostname is desired since it allows a non-responsive secondary Application Server to be successfully bypassed with the same benefit of traversing the secondary Application Server proxy. The default value is nil. If the value is nil and IPv4 is supported, publicIPAddress is used if not nil. If nil, "localhost" is resolved for use. If the value is nil and only IPv6 is supported, publicIPv6Address is used if not nil. If nil, a known IPv6 address is used.
bw.sip.accessrecordrouteincludetcptransport	When set to "true", this parameter indicates to include the SIP-URI transport value TCP when adding or rewriting the Record-Route entry and sending the request or response over TCP to an access-side device. The default value is "true".
bw.sip.accessrecordrouteincludeudptransport	When set to "true", this parameter indicates to include the SIP-URI transport value UDP when adding or rewriting the Record-Route entry and sending the request or response over UDP to an access-side device. The default value is "false".
bw.sip.accessrecordrouteport	This parameter is used by the secondary Application Server proxy to populate a Record-Route entry's port if bw.sip.accessrecordroutehost is not nil. The default value is nil, which indicates that the port should not be sent. If the value is not nil, it should be set to the configured SIP listeningPort, (which defaults to "5060"); however, this prevents the sender from performing a NAPTR and SRV lookup. The default value is nil. If bw.sip.accessrecordroutehost is nil, the configured SIP listeningPort is used.
bw.sip.networkrecordroutehost	This parameter has the same meaning as the

PARAMETER	DESCRIPTION AND ALTERNATIVES
	bw.sip.accessrecordroutehost; however, this parameter is applicable to network-side devices. Populating with the reverse cluster network-side hostname is desired since it allows a non-responsive secondary Application Server to be successfully bypassed with the same benefit of traversing the secondary Application Server proxy. The default value is nil. If the value is nil and IPv4 is supported, then privateIPAddress is used (if not nil). If privateIPAddress is nil, then the publicIPAddress is used (if not nil). If nil, “localhost” is resolved for use. If the value is nil and only IPv6 is supported, privateIPv6Address is used (if not nil). If privateIPv6Address is nil, then the publicIPv6Address is used (if not nil). If nil, a known IPv6 address is used.
bw.sip.networkrecordrouteincludetcptransport	This parameter has the same meaning as the bw.sip.accessrecordrouteincludetcptransport; however, this parameter is applicable to network-side devices. The default value is “true”.
bw.sip.networkrecordrouteincludeudptransport	This has the same meaning as the bw.sip.accessrecordrouteincludeudptransport ; however, this parameter is applicable to network-side devices. The default value is “false”.
bw.sip.networkrecordrouteport	This parameter has the same meaning as the bw.sip.networkrecordroutehost; however, this parameter is applicable to the peer network interface. If set, this parameter should be a network-side IP-address or hostname corresponding to the peer network interface to this Application Server instance. The default value is nil. If the value is nil and IPv4 is supported, then the privateIPAddress is used (if not nil). If privateIPAddress is nil, then the publicIPAddress is used (if not nil). If nil, “localhost” is resolved for use. If the value is nil and only IPv6 is supported, then the privateIPv6Address is used (if not nil). If the privateIPv6Address is nil, then the publicIPv6Address is used (if not nil). If nil, a known IPv6 address is used.
bw.sip.peernetworkrecordrouteincludetcptransport	This parameter has the same meaning as the bw.sip.networkrecordrouteincludetcptransport ; however, this parameter is applicable to the peer network interface. The default value is “true”.

PARAMETER	DESCRIPTION AND ALTERNATIVES
bw.sip.peernetworkrecordrouteincludeudptransport	This parameter has the same meaning as the bw.sip.networkrecordrouteincludeudptransport; however, this parameter is applicable to the peer network interface. The default value is “false”.
bw.sip.peernetworkrecordrouteport	This parameter has the same meaning as the bw.sip.networkrecordrouteport; however, this parameter is applicable to the peer network interface. The default value is nil.

In the example used in this section, the default values for these parameters are adequate and therefore do not have to be explicitly changed.

When changing the values used to populate the Record-Route entry’s SIP-URI host (bw.sip.accessrecordroutehost, bw.sip.networkrecordroutehost, and bw.sip.peernetworkrecordroutehost or their alternatives), aliases for the secondary Application Server or Application Server cluster must be used. The alias list can viewed and modified through the secondary Application Server CLI under `AS_CLI/System/Alias`.

6.1.18.4 Peer SIP Connectivity Monitoring

Peer SIP connection monitoring can be enabled and controlled using the following values available under `AS_CLI/System/Redundancy/PeerSIPConnectionMonitoring:`

`enabled` – If this parameter has a “true” value, SIP connectivity monitoring is enabled. The default value is “false”.

`heartbeatIntervalInMsec` – This parameter controls the interval between OPTIONS requests. The default value is “1000” (milliseconds).

`heartbeatTimeoutInMsec` – This parameter controls the timeout value for the sending Application Server to receive a response to the OPTIONS request. The default value is “5000” (milliseconds).

```
AS_CLI/System/Redundancy/PeerSIPConnectionMonitoring> get

  enabled = false

  heartbeatIntervalInMilliseconds = 1000

  heartbeatTimeoutInMilliseconds = 5000
```

If the Application Server uses the same network interface for SIP messages as for the redundancy link, then the monitoring of the redundancy link is sufficient, and SIP connectivity monitoring should not be enabled, as it is the case in the example used in this section.

All these parameters can be configured during the installation or using the UNIX command `setup-redundancy`.

ELEMENT MANAGEMENT SYSTEM CONFIGURATION

This section outlines the EMS server configuration steps related to Clearspan redundancy.

6.1.19 FAILOVER PARAMETER CONFIGURATION

The failover functionality makes use of a heart beat mechanism where the active server informs the standby server it is alive. The rate at which the active server notifies the standby server is determined by the value of the parameter HEART_BEAT_INTERVAL. For instance, when this parameter is set to 20 seconds, the active server notifies the standby server every 20 seconds. At every FAIL_OVER_INTERVAL, the standby server verifies that it has received a notification from the active server. The value of the parameter FAIL_OVER_INTERVAL must be greater than the value of the parameter HEART_BEAT_INTERVAL to avoid false active server failure detection. For the active server to be declared failed, there must be a certain number of FAIL_OVER_INTERVALs, during which the standby server has not received a notification. The number of FAIL_OVER_INTERVALs is determined by the parameter RETRY_COUNT. The following table presents the default value for the parameters.

TYPE	DEFAULT VALUE
HEART_BEAT_INTERVAL	20
FAIL_OVER_INTERVAL	30
RETRY_COUNT	2

The time during which the EMS functionality is unavailable (except for receiving traps) is called the failover period. This period can be determined using the following formula.

Failover period = (FAIL_OVER_INTERVAL * RETRY_COUNT) + server startup time

All these parameters can be configured during the installation or using the UNIX command `setup-redundancy`.

PROFILE SERVER CONFIGURATION

The redundancy for the Profile Server is configured using the script `config-redundancy`. This script must be run on all servers.

CLUSTER DATA REPLICATION MANAGEMENT

There are two underlying mechanisms responsible for data replication on the servers. The first one, RSYNC is responsible for replicating a pre-defined list of files on the system, including the voice portal prompts and greetings. The second one, TimesTen Replication is responsible for replicating the end-user information between the server databases.

Both mechanisms can be started, stopped, and monitored from either the CLI or UNIX prompt.

6.1.20 TIMESTEN REPLICATION DAEMON

For database replication to work properly, the TimesTen database replication daemon must be running. Replication is automatically started when upgrading to Release 10.0 or for a fresh install. Replication also restarts automatically for reboots of the server.

WARNING: Replication should never be manually stopped unless expressly required as part of a maintenance procedure. Manually stopping TimesTen Replication on any server can result in a cluster database out-of-sync condition.

Note: Performing a `stopbw` does not stop replication.

```
***** Clearspan CLI *****
AS_CLI/System/Peering> status
Redundancy/Replication Status
-----------------------------
File Replication pid = 1289
Datastore name = NetworkServer
Replication Agent Policy        : always


***** Unix Prompt *****
as1$ repctl status
Redundancy/Replication Status
-----------------------------
File Replication pid = 1289
Datastore name = AppServer
Replication Agent Policy        : always
```

When the data replication has started, an operator should expect:

- A file replication `pid` to indicate that RSYNC is running.
- The replication agent policy is set to always indicate that TimesTen replication is started and then re-started on subsequent system reboots.

6.1.21 MYSQL REPLICATION (EMS)

For database replication to work properly, the MySQL database replication must be configured. Replication is automatically configured and started when installing and configuring the EMS or when running the UNIX script `setup-redundancy`. Replication also restarts automatically for reboots of the server.

WARNING: Reeplication should never be manually stopped unless expressly required as part of a maintenance procedure. Manually stopping MySQL Replication on any server can result in a cluster database out-of-sync condition.

Note: PERFORMING A `STOPBW` **DOES NOT STOP REPLICATION.**

```
***** Unix Prompt *****
bwadmin@mtlems05$ repctl status

Redundancy/Replication Status
-----------------------------
File Replication pid(s) =  1072
Datastore name = WebNmsDB
Replication Agent Policy        : always

Replication State
mtlems05: (filerep: true)(database: true)
mtlems06: (filerep: true)(database: true)

Database Replication Lagging State
mtlems05: (false) Master/Slave position: 361403185/361403185 (delta: 0)
mtlems06: (false) Master/Slave position: 327508640/327508640 (delta: 0)
```

6.1.22 CLUSTER PEER MEMBERS

Cluster peer member information can be accessed using the server's CLI or UNIX. An operator can also lock the local database or the database of a peer using the CLI to prevent modifications during a maintenance window.

WARNING: A database should not be locked unless expressly required as part of a maintenance procedure. Most Clearspan procedures (for example, software upgrades) automatically manage database locking as required. Manually locking a database does not affect basic call processing, but database information cannot be modified (that is, users cannot change their service profile.

```
**** Clearspan CLI *****
NS_CLI/System/Peering/Peers> ?
    0)         get : show Network Server peer-related attributes
    1)         add : add a new Network Server peer
    2)      delete : delete an existing Network Server peer
    3)        lock : to lock the database
    4)       start : to start database replication
```

```
    5)      status : to obtain status of system replication
    6)        stop : to stop database replication
    7)      unlock : to unlock the database

NS_CLI/System/Peering/Peers> get
HostName                 Address                                State
==================================================================
  MTLNS01                 MTLNS01.int.domain.com          unlocked
  MTLNS02                 MTLNS02.int.domain.com          unlocked

***** Unix Prompt *****
as1$ IHApp$ peerctl ls
HOSTNAME/ADDRESS        State
-------------------------------------------------------
as2/as2                 unlocked
as1/as1             primary,unlocked
2 entries found
```

6.1.23 MONITOR DATA REPLICATION

Data replication can be automatically monitored via the `healthmon` utility. The `healthmon` utility performs a system-level check on all Clearspan subcomponents and can be configured to periodically send SNMP traps. With respect to data replication, `healthmon` replicates test data in both directions to ensure replication is functioning properly.

Along with being configured for automatic keep-alive, the `healthmon` utility should be run on all cluster peers after any Clearspan maintenance procedure, to ensure replication is functioning properly.

```
IHApp$ healthmon -l -d
--------------------------------
System Health Report Page
   Clearspan Server Name: IHApp
   Date and time         : Mon May 19 15:11:35 EDT 2003
   Report severity       : NOTIFICATION
   Server type           : AppServer
--------------------------------
Loading configuration
Configuration loaded

Validating the file system size using WARNING(80%), HIGH(90%),
CRITICAL(95%) and STOPBW(98%) as the threshold values
Testing partition /
Testing partition /tmp
Testing partition /bw
Filesystems all under threshold value

Validating that all processes are started for the AppServer application
 Monitoring Application Server processes + Apache, Tomcat, tnameserv
... Monitoring TimesTen process
... Monitoring database replication
... Monitoring file replication
All AppServer required processes are running

Validating that all SNMP subagents are running (MIBIISA, ESD)
All subagents are running

--------------------------------
```

ID UNIQUENESS ACROSS APPLICATION SERVER CLUSTERS

When deploying multiple Application Server clusters, the following restrictions for uniqueness apply to system object IDs:

- Enterprise IDs must be unique across all Application Server clusters using the same Network Server.
- Service provider IDs can be duplicated on different Application Server clusters, as they are not synchronized on the Network Server.
- Group IDs can be duplicated on different Application Server clusters. Although they are synchronized on the Network Server, they are stored along with their hosting enterprise or service provider.
- User IDs (of the format *ID@domain*) must be unique across all Application Server clusters using the same Network Server

7 INDEX

Clearspan®