TECHNICAL REFERENCE

May 2021 | 3725-49225-001A

# Solution Guide

## Failover and Route Redundancy

**GETTING HELP**

For more information about installing, configuring, and administering Poly/Polycom products or services, go to the Poly Online Support Center.

Plantronics, Inc.
345 Encinal Street
Santa Cruz, California
95060



**PLANTRONICS + POLYCOM**
**NOW TOGETHER AS POLY**

# Contents

# DNS Based Route Redundancy on Polycom Phones

This document is supplementary to the information contained in these public guides:

- https://support.polycom.com/content/dam/polycom-support/products/voice/polycom-uc/other-documents/en/configuring-optional.pdf
- https://support.polycom.com/content/dam/polycom-support/products/voice/polycom-uc/other-documents/en/2012/sip-server-fallback-tb5844.pdf

The following definitions should be understood in the context of this document:

- **Failover** - The act of moving a registration and associated SIP subscriptions from one SBC to a different SBC as the result of a detected failure of the original SBC. This failure may be transitory in nature or persistent.
- **Failover state** - A line is in a failover state whenever it is not registered to the primary SBC
- **Failback** - When in the failover state, the act of moving a registration from the current operational SBC to an SBC with a higher priority/weight in the same SRV record after expiry of the configured failback duration interval.
- **Primary SBC** - Once all DNS NAPTR/SRV/A queries complete, the phone will create an ordered list of each route based on the Order, Preference, Priority, and Weights of the returned records. Once the phone successfully registers with an SBC within an SRV record, the SBC within that same SRV record with the lowest numerical priority is the Primary SBC. Records with equal Priority are allocated into the ordered list at the time of its initial creation based on the weight.

## Configuring the Server Address

Poly phones use DNS records to provide a list of routes available for request handling. Both A records that resolve to two or more IPs, or SRV records that resolve to two or more A records can be used in this manner.

In order to query an address as an SRV record, there must be no explicit port defined in configuration. This means setting the below:

```
voIpProt.server.1.address="example.polycom.com"
voIpProt.server.1.port="5060"
```

does **NOT** send an SRV query as the port is already defined. The default value for server port is 0 (interpreted as unknown) meaning the following will allow for DNS SRV-based redundancy to work.

```
voIpProt.server.1.address="example.polycom.com"
voIpProt.server.1.port="0"
or
voIpProt.server.1.port=""
```

Given the preceding parameters, a phone will first query `example.polycom.com` via SRV. If there's no record returned, it will then also query the same address as an A record and assume a port of 5060.

Once DNS queries are complete, the phone will order its received list of servers/routes based on the weight and priority returned by the SRV record, or by the order specified in the A record response. You can obtain lists of up to four addresses in this manner.

For example, assume the SRV query for `_sip._udp.example.polycom.com` resolves as follows:

```
_sip._udp.example.polycom.com
          priority       = 10
          weight         = 0
          port           = 5060
          svr hostname   = summerland.polycom.com
_sip._udp.example.polycom.
          priority       = 20
          weight         = 0
          port           = 5060
          svr hostname   = kelowna.polycom.com
_sip._udp.example.polycom.
          priority       = 30
          weight         = 0
          port           = 5060
          svr hostname   = naramata.polycom.com
summerland.polycom.com     internet address = 10.242.16.2
nakusp.polycom.com            internet address = 10.149.16.9
naramata.polycom.com    internet address = 192.17.51.43
```

The phone will use the Summerland server as the primary with two additional failover servers available should Summerland become unresponsive.

# Registration Periods and Overlap

The phones will keep a configurable overlap period where re-registrations are attempted before the full expiry of the registration. The default timers are a 1-hour registration with a 60 second overlap.

| Parameter | Values (In Seconds) | Default |
|---|---|---|
| reg.X.server.Y.expires | 10 to 2147483647 | 3600 |
| reg.X.server.Y.expires.overlap | 5 to 65535 | 60 |
| reg.X.server.Y.subscribe.expires | 10 to 2147483647 | 3600 |
| reg.X.server.Y.subscribe.expires.overlap | 5 to 65535 | 60 |
| voIpProt.server.X.expires | 10 to 2147483647 | 3600 |
| voIpProt.server.X.expires.overlap | 5 to 65535 | 60 |
| voIpProt.server.X.subscribe.expires | 10 to 2147483647 | 3600 |
| voIpProt.server.X.subscribe.expires.overlap | 5 to 65535 | 60 |

# DNS Ordering Changes While a Phone is Registered

Poly phones that have already received a route set and registered, and which aren't currently in a failover state, will not react to reordering of A records or changes to weight/priority of SRV records if the route to

the current registrar is still present in the newly returned route set. Put another way, the current route is "sticky" until it's removed or a network error is detected.

If the phone receives a DNS response for its current route and the IP address that is currently in use is no longer listed, then the phone will immediately attempt to re-register with the new ordering.

If a phone that is currently registered has received an updated DNS ordering and then experiences a network error triggering failover, then it will iterate through the last received list of records in order of weight/priority.

# DNS NAPTR Records for Mixed Transports Across Multiple Networks

Available as of UC Software 6.3.1 & 6.4 or later.

A NAPTR query may receive a response containing service records (SRV) for several transport protocols (UDP, TCP, or TLS). In such cases, the Order and Preference values of the NAPTR records are used to rank the protocol choices and once a registration is made from within a service record, the phone will attempt all SBCs in that service record before moving to the next ranked service record.

Assume the following:

| | NAPTR | Order | Pref | SRV | Priority | Weight | Acess IP | Selection Order |
|---|---|---|---|---|---|---|---|---|
| a. TCP | poly.example.com | 50 | 10 | _sip._tcp.poly-sbc.example.com. | 100 | 50 | 200.1.1.5 | 1 |
| | | | | _sip._tcp.poly-sbc.example.com. | 100 | 50 | 200.2.2.9 | 1 |
| b. TCP | poly.example.com | 100 | 10 | _sip._tcp.poly-sbc.example.com. | 100 | 50 | 192.168.0.50 | 2 |
| | | | | _sip._tcp.poly-sbc.example.com. | 100 | 50 | 192.168.0.100 | 2 |
| c. TLS | poly.example.com | 150 | 10 | _sips._tls.poly-sbc-ext.example.com. | 140 | 50 | 192.148.100.5 | 3 |
| | | | | _sips._tls.poly-sbc-ext.example.com. | 240 | 50 | 192.148.100.20 | 4 |
| | | | | _sips._tls.poly-sbc-ext.example.com. | 340 | 50 | 192.148.200.4 | 5 |
| | | | | _sips._tls.poly-sbc-ext.example.com. | 440 | 50 | 192.148.200.30 | 6 |
| d. UDP | poly.example.com | 200 | 10 | _sip._udp.poly-sbc-bak.example.com. | 140 | 50 | 192.168.50.6 | 7 |
| | | | | _sip._udp.poly-sbc-bak.example.com. | 240 | 50 | 192.168.50.64 | 8 |

The "Selection Order" in the far-right column is the way a phone would rank all possible routes returned by this NAPTR record set. Where *Selection Order* is the same due to identical priority and weight, the phone will randomize the selection based on the weights at the time of resolving the record so that any 2 phones may have small variations in their final ordered list.

Assume further that SBCs representing *Selection Order* 1 & 2 are not routable in the network the phone is plugged into and that SBCs representing selection order 3 & 4 are currently down. The first SBC the phone can register to in this scenario is *Selection Order=5*. This phone is considered to be in the failover state since the primary SBC is the lowest priority SBC for its currently registered service record (*Selection Order=3).* The phone will attempt to *failback* to SBC 3 periodically if configured for failback. Should SBC 5 fail, the phone will failover and try SBCs in order starting from the top of its service record (the primary SBC) but skipping its current and known failed SBC, so for our example, SBC 5 fails, the phone will try SBC 3, 4, 6, then move to a new service record for SBC 7 where if it registers, the phone will now treat SBC 7 as the new primary.

# Re-Registration on Failover (RROFO)

RROFO offers additional flexibility and control to a Poly phone's redundancy behavior allowing you to:

- Register to a failover server before sending a request
- Ignore signaling from any server other than the currently registered server
- Remain on a failover server for a set period of time before failback occurs
- Unregister with the current route before the failback to a primary server

## Triggering Failover

Failover is triggered by failure to complete a SIP request response pair. This can take several forms:

- **ICMP error:** receiving most ICMP errors will immediately trigger a failover.
- **SIP error:** response code 503 can be optionally configured as a failover trigger using **`voIpProt.SIP.failoverOn503Response`**. By default, this is enabled. All other SIP response codes will not generate a failover.
- **No response:** failure to receive response to an ARP, TCP SYN, or TCP/UDP request in the configured period of time.

Whereas **ICMP errors** and **SIP errors** will trigger immediate failover, **no responses** must determine the difference between network delay, lost packets, and an unavailable network element. To do this, multiple request attempts are made before marking the current server as unavailable.

**`voIpProt.SIP.failoverOn503Response`**

0 - A 503 response will not trigger a failover. It will be treated as a final response to the current SIP dialog. If a retry-after header is present, the value specified will be used before a new request dialog is attempted again.

1 (Default) - All SIP requests receiving a 503 response will immediately trigger failover. A retry-after header in a 503 response is only applied when sending a request through the last failover route.

## When Failover is Not Triggered

Certain errors or connection issues don't trigger failover and instead are treated as a registration loss. These will prompt the phone to "back-off" for a randomized period of time before attempting to re-register. See the section **Controlling the randomized back-off period between registrations** for more info on altering the randomized period.

- TCP stream drop from the far end (either a FIN or a RST)
- 403 or any other denial response to any SIP request other than a 503
- Failure to receive a keep-alive or a keep-alive acknowledgement

## *Failover Limitations of SIP Response Codes*

Failover is a system primarily able to detect issues when sending *requests* by the phone and not when sending *responses*. When sending responses over UDP where reliable delivery is not guaranteed, the phone will never know that the response did not arrive, making failover on responses impossible.

Similarly, when the phone responds to a SIP request, if the response is delivered over TCP/TLS and receives no TCP ACK confirming reliable delivery of the packet, then failover for delivery of the SIP response *will not occur*.

# Fine Tuning Failover Timings (UDP)

The following parameters control SIP timers B & F which play a role in the failover speed between routes and when a transaction is determined cancelled.

| Parameter | Values | Default | Notes |
|---|---|---|---|
| voIpProt.server.1.retryMaxCount | 0 to 65535 | 3 | |
| voIpProt.server.1.retryTimeOut | 0 to 65535 (milliseconds) | 0 | |
| voIpProt.SIP.tcpFastFailover | 0 or 1 | 0 | |
| voIpProt.SIP.tcpFastFailover.timeout | 2000 - 5000 | 5000 | New in UCS 6.3.1 & 6.4 or later |

A **retryTimeOut** setting of Zero ("0"), as in **voIpProt.server.1.retryTimeOut="0"**, is a special case where the phone will use the RFC 3261 backoff timers (0.5s, 1s, 2s, 4s, 4s, 4s, etc.) until a complete 32 second transaction timer expiry has been reached. How many of the backoff steps are used depends on the value of the **retryMaxCount**.

The preceding settings allow adjustment of failover speed from route to route until the final route in the list is reached. The final route is repeatedly tried for whatever time remains of the 32 second transaction period since the original request to the primary SBC began.

**Example 1 - Using the Poly default settings, transport = UDP.**

- The initial attempt (counter == 1)
  - Delay: 0.5 seconds
- 1st retry (counter == 2)
  - Delay: 1 second
- 2nd retry (counter == 3, retries complete)
  - Delay 2 seconds
- Failover to next server (or randomized back off period and try again if no additional servers are configured)

Total delay before failover to next server is 3.5 seconds

**Example 2 - static timing between retries, transport = UDP**

reg.1.server.1.address="domain.com" → Assume this resolves to 3 A records
reg.1.server.1.retryTimeOut="5000"
reg.1.server.1.retryMaxCount="2"

- 1st server will be tried twice with a wait period of 5 seconds after each attempt. (10 seconds total)
- Failover to the 2nd IP. Try 2 times with wait period of 5 seconds (10 seconds total)
- Failover to the final (3rd) server, retries continue for remaining 32 - 20 = 12 seconds with intervals defined by RFC 3261 (0.5s, 1s, 2s, 4s, 4s, 4s)

# Understanding TCP/TLS Failover Timeouts

Because TCP is a reliable transport, no retransmissions are made unless a TCP ACK is never received. Any SIP application layer request that receives a TCP ACK but remains unanswered is expected to wait until the full 32 second transaction timer expires. This behavior is overridden by **`voIpProt.SIP.tcpFastFailover`**, allowing TCP to adhere to a faster timeout.

As of UCS 6.3.1 & UCS 6.4, TCP failover timing can be controlled by setting **`voIpProt.SIP.tcpFastFailover.timeout`** where the value in milliseconds is the wait period before failing the current route to the next. This new timeout value only applies when the **`retryTimeOut="0"`**

In earlier UC Software versions, the TCP failover delay is based on multiplying **`voIpProt.server.1.retryMaxCount`** and **`voIpProt.server.1.retryTimeOut`** settings together to create a wait period that would mimic the same retry delay that UDP would use.

#### Example 3 – TCP using the pre-UCS6.3.1 fastFailover algorithm

```
voIpProt.server.1.retryMaxCount="4"
voIpProt.server.1.retryTimeOut="500"
voIpProt.SIP.tcpFastFailover="1"
voIpProt.SIP.tcpFastFailover.timeout="4000"     (ignored since retryTimeOut
doesn't =0)
```

- The initial attempt (counter == 1)
- Delay: retryMaxCount * retryTimeOut = 4 * 500 = 2 seconds
- Failover to next server

#### Example 4 – TCP using a fastFailover timeout

```
voIpProt.server.1.retryMaxCount="4"       (ignored)
voIpProt.server.1.retryTimeOut="0"  (enables the fastfailover.timeout
setting)
voIpProt.SIP.tcpFastFailover="1"
voIpProt.SIP.tcpFastFailover.timeout="4000"
```

- The initial attempt (counter == 1)
- Delay: 4 seconds (the value of voIpProt.SIP.tcpFastFailover.timeout is used)
- Failover to next server

## *Failover and TLS require Keep-Alives*

When using with TLS, a TCP stream is required to be maintained for continued communication between the phone and its server. Keep-alives maintain the required TCP connection and the following parameters govern how to control the timing of those keep-alives.

Note that a lack of response to a sent keep-alive is not a trigger for failover as the keep-alives in this context are used exclusively for keeping firewall ports open and the TCP stream in place.

| Parameter | Values | Default |
|---|---|---|
| **tcpIpApp.keepalive.tcp.sip.tls.enable**<br>Enable or disable TLS keep-alives | 0 or 1 | 0 |
| **tcpIpApp.keepalive.tcp.noResponseTransmitInterval**<br>How long to wait between keep-alive retransmit attempts. The phone must make 4 attempts before declaring the connection lost. | 5 to 120 | 20 |
| **tcpIpApp.keepalive.tcp.idleTransmitInterval**<br>How often a new keep-alive is sent when the phone is idle | 10 to 7200 | 30 |
| **tcpIpApp.keepalive.tcp.sip.persistentConnection.enable**<br>When enabled, the phone will acknowledge far end initiated keep-alives and not drop the TCP connection when idle. If set to 0, provided no keep-alives are initiated by the phone and even if far end keep-alives are being received, the phone will close its TCP connection after 60 seconds of inactivity. | 0 or 1 | 0 |

## *UDP & TCP Keep-Alives*

Ideally UDP & TCP routes are maintained through SIP registration refreshes but aggressive firewall rules may necessitate a faster keep-alive rate to keep the firewall port open for inbound calls.

| Parameter | Values | Default |
|---|---|---|
| **nat.keepalive.interval**<br>Enable or disable UDP keep-alives. Keep-alives are disabled when set to 0. Any non-zero integer refers to the rate in seconds that a keep-alive is transmitted. | 0 – 3600 (seconds) | 0 |
| **nat.keepalive.udp.payload**<br>A configurable payload transmitted in the data portion of the keep-alive. | String, max 255 chars | CRLFCRLF |
| **nat.keepalive.tcp.payload**<br>A configurable payload transmitted in the data portion of the keep-alive. | String, max 255 chars | CRLFCRLF CRLFCRLF CRLF |

# Failback

Failback is the process of returning registration and subscriptions to the primary server. A SIP registration will not attempt failback while a call is active on that registration and will instead continue to renew its registration with the current SBC until the call completes, at which point, failback will immediately occur.

If Enhanced Failover is enabled, then failback will not occur while any SIP registration has an active call. Once all calls are cleared, all lines will failback to the primary SBC.

By default, failback attempts occur for each new SIP request but this may be changed to wait a set duration or to not failback at all using the **`failback.mode`** parameter.
When failback is set to its default, note that an authentication challenge (SIP response code 401 or 407) prompts a new request and so the phone will retry its primary which can further increase delays. This authentication on failback step can be skipped by setting **`voIpProt.SIP.authOptimizedInFailover="1".`**

**`voIpProt.SIP.authOptimizedInFailover`**

0 (default) - The first new SIP request is sent to the server with the highest priority in the server list when failover occurs.

1 - The first new SIP request is sent to the server that sent the proxy authentication request when failover occurs.

**Example:**

A phone has registered to its primary server and has an expiry of 3600 seconds. 10 minutes after registration, the primary server becomes unavailable and the user needs to place a call.

Assuming default settings, the phone will try its primary three times as described in the "Triggering Failover" section before failing over to address #2 where presumably the call succeeds. A second call placed by that user will go through the same failover sequence, as will the next re-registration attempt once its expiry timer fires. These failover and fallback sequences will persist as long as the primary server is unavailable.

If this constant moving back and forth isn’t desirable, see the on **Re-Registration on Failover Configuration** section to either enable RROFO or to change the failback mode to a setting other than "newRequest".

# Subscriptions and RROFO

All previously existing Subscriptions (BLF, BLA, SCA, ACD, etc.) are invalidated once a REGISTER to a new server/SBC is successfully accepted which will then immediately trigger new subscriptions to be started with the newly registered server.

It’s important to recognize that when RROFO is NOT in use, that other failover methods will not restart a subscription immediately.

# SIP Call-ID Behavior on Failover, Failback, and When Unregistered

When a call or other SIP dialog such as a registration request determines that the primary route or SBC is unavailable and attempts to failover to another route, the SIP Call-ID header value will not change. This is because the dialog hasn't yet terminated and the phone is trying other possible paths to route the request.

The same applies when failing back from a failover path and returning to the primary SBC/route and so the phone will continue to use the same Call-ID rather than create a new one.

The only time a new call-ID is created is when the phone has transitioned to an unregistered state. The unregistered state can occur due to a failure to locate an SBC/route or in cases where the TCP/TLS stream is terminated causing immediate transition.

See the below section **When All Failover Routes Fail** for more details.

# When All Failover Routes Fail

If the phone has attempted to failover to every address returned by DNS without succeeding, a back-off period is started once the full 32 second period has elapsed against the last server in the failover pool.

This back-off period is random and is behaves in the same manner as would a failed registration where no additional failover servers are defined. The back-off is designed to randomize the requests so that in cases where there are power outages or loss of LAN, that large phone installations don't constantly flood servers.

When the randomized backoff period has elapsed, the phone will attempt to register again, and will use a new SIP Call-ID unless otherwise configured based on **voIpProt.SIP.newCallOnUnRegister**.

**voIpProt.SIP.newCallOnUnRegister**

0 - New REGISTER messages will reuse the last SIP Call-ID from before the phone lost registration.

1 (default) - After losing registration completely and exhausting all possible failover routes, new REGISTER messages will be considered as a new SIP Dialog and generate a new Call-ID.

This parameter does NOT apply to messages sent as part of failover/failback routing.

# Controlling the randomized back-off period between registrations

Back-off period control was added in UCS 4.1.0 using the calculations provided in RFC 5626. Before this change, back-off was a random time between 30 and 60 seconds. It's important to note that although we implemented this additional optional RFC based back off control, we don't claim full support of RFC 5626 and the notion of signaling "flows".

| Parameter | Values | Default |
|---|---|---|
| reg.x.server.y.registerRetry.baseTimeOut | 10 to 120 | 60 |
| reg.x.server.y.registerRetry.maxTimeOut | 60 to 1800 | 60 |
| voIpProt.server.X.registerRetry.baseTimeOut | 10 to 120 | 60 |
| voIpProt.server.x.registerRetry.maxTimeOut | 60 to 1800 | 60 |

Set **registerRetry.maxTimeOut** with default="60" to preserve our longstanding behavior of reconnect attempts between 30-60 seconds. You can go larger as the RFC describes but eventually you start backing off so long that the endpoint is typically seen as malfunctioning rather than just complying with its intended design.

Set **registerRetry.baseTimeOut="10"** to greatly shorten the back-off time to 5-10 seconds.

# Failover and Emergency Dialing for Shared (SCA) Lines

Use cases where only a shared line exists on a phone.

If the server/SBC is unavailable, the line seize requests will continue trying across 10 retransmissions or until 32 seconds has elapsed. At that time the call passes to a later stage of SIP stack processing and the stack becomes aware that it's an emergency call and sends anyways. Decrease the line-seize attempt using:

**`voIpProt.SIP.lineSeize.retries`**

10 (Default)
3 to 10

# RROFO Configuration

| Parameter | Values | Default |
|---|---|---|
| voIpProt.server.x.failOver.reRegisterOn<br>voIpProt.SIP.outboundProxy.failOver.reRegisterOn<br>reg.x.server.y.failOver.reRegisterOn<br>reg.x.outboundProxy.failOver.reRegisterOn | 0 or 1 | 0 |
| voIpProt.server.x.failOver.onlySignalWithRegistered<br>voIpProt.SIP.outboundProxy.failOver.onlySignalWithRegistered<br>reg.x.server.y.failOver.onlySignalWithRegistered<br>reg.x.outboundProxy.failOver.onlySignalWithRegistered | 0 or 1 | 1 |
| voIpProt.server.x.failOver.failRegistrationOn<br>voIpProt.SIP.outboundProxy.failOver.failRegistrationOn<br>reg.x.server.y.failOver.failRegistrationOn<br>reg.x.outboundProxy.failOver.failRegistrationOn | 0 or 1 | 1 |
| voIpProt.server.x.failOver.failBack.mode<br>voIpProt.SIP.outboundProxy.failOver.failBack.mode<br>reg.x.server.y.failOver.failBack.mode<br>reg.x.outboundProxy.failOver.failBack.mode | duration<br>newRequests<br>DNSTTL<br>registration | duration |
| voIpProt.server.x.failOver.failBack.timeout<br>voIpProt.SIP.outboundProxy.failOver.failBack.timeout<br>reg.x.server.y.failOver.failBack.timeout<br>reg.x.outboundProxy.failOver.failBack.timeout | 0, 60-65535 | 3600 |
| voIpProt.server.x.failOver.unRegisterOnFailBack<br>voIpProt.SIP.outboundProxy.failOver.unRegisterOnFailBack<br>reg.x.server.y.failOver.unRegisterOnFailBack<br>reg.x.outboundProxy.failOver.unRegisterOnFailBack | 0 or 1 | 0 |
| voIpProt.SIP.outboundProxy.failOver.enhanced.enable | 0 or 1 | 0 |

**reRegisterOn**

This parameter enables/disables the **Re-Registration On Failover** feature and must be enabled for any of the below parameters to be considered.

1 - The phone attempts to register to any server or SBC before continuing with any other pending SIP request. If the registration succeeds (a 200 OK response with valid expires), signaling will proceed with that server/SBC.

0 - The phone does not guarantee a new registration is in place and will try each new request in order of priority across the list of failover servers/SBCs.

**onlySignalWithRegistered**

1 - No signaling is accepted from or sent to a SBC that doesn't hold the currently active registration. If the phone needs to send signaling associated with an existing call via an unregistered SBC (for example, to resume or hold a call), the call will end. No SIP messages will be sent to the unregistered SBC.

0 - Signaling will be accepted from and possibly sent to a SBC has no active registration.

**failRegistrationOn**

1 - The phone will silently invalidate an existing registration (if it exists) along with all linked subscriptions, at the time failover begins.

0 - Existing registrations will remain active until their expiries lapse normally. This means that the phone will attempt failback without first attempting to register to the primary SBC to determine if it has recovered.

**failBack.mode**

newRequests - All new requests are forwarded first to the primary SBC regardless of the last used SBC.

DNSTTL - The phone tries the primary SBC again after a timeout equal to the DNS TTL configured for the SBC that the phone is registered to.

Duration - The phone tries the primary SBC again after the time specified by **reg.x.outboundProxy.failOver.failBack.timeout** expires.

**failBack.timeout**

The time to wait (in seconds) before failback occurs (overrides **reg.x.server.y.failOver.failBack.timeout**).

Duration - The phone waits this long after connecting to the current working SBC before selecting the primary SBC again.

0 - The phone will not fail-back until a failover event occurs with the current SBC.

**unRegisterOnFailBack**

New in UCS 6.3.0 and later software.

1 - Before failing back to the primary SBC, send a REGISTER with expires:0 to unregister from the current SBC.

**failOver.enhanced.enable**

New in UCS 6.3.1 and later software.

Assume an SRV record returns 8 A records representing possible signaling routes.

0 - A transaction will have a maximum of 32 second life unless timer B is reached based on the configuration of the retryMaxCount and retryTimeout parameters. Each of the possible routes will be tried in sequence until timer B is reached.

1 - The delay between trying each route remains configurable by the retryMaxCount and retryTimeout parameters or by the voIpProt.SIP.tcpFastFailover.timeout parameter, but timer B will not be used to stop the transaction before every route has been attempted.

# Example Failover Configurations

## Best Practice: UDP signaling with two SBCs using RROFO

Assume a single FQDN that resolves via DNS SRV to 2 A records representing geo-redundant SBCs.

```
reg.1.address="1000"
voIpProt.server.1.address="server.coreSide.com"
voIpProt.server.1.retryTimeOut="500"
voIpProt.server.1.retryMaxCount="3"
voIpProt.SIP.tcpFastFailover="1"
voIpProt.SIP.authOptimizedInFailover="1"
voIpProt.SIP.failoverOn503Response="1"
voIpProt.SIP.outboundProxy.address="sbc.accessSide.com"
voIpProt.SIP.outboundProxy.port="0"
voIpProt.SIP.outboundProxy.transport="DNSnaptr"
voIpProt.SIP.outboundProxy.failOver.reRegisterOn ="1"
voIpProt.SIP.outboundProxy.failOver.onlySignalWithRegistered="1"
voIpProt.SIP.outboundProxy.failOver.failBack.mode="duration"
voIpProt.SIP.outboundProxy.failOver.failBack.timeout="600"
voIpProt.SIP.outboundProxy.failOver.failRegistrationOn="1"
```

## Best Practice: TLS Signaling with two SBCs using RROFO and Keep-Alives

Assume a single FQDN that resolves via DNS SRV to 2 A records representing geo-redundant SBCs. The phone will initiate keep-alives to maintain the TCP stream and should the stream be disconnected; we've overridden the standard back-off period from a random 30-60 seconds down to 5-10. Differences from the UDP example above are illustrated in `BLUE TEXT`.

```
reg.1.address="1000"
voIpProt.server.1.address="server.coreSide.com"
voIpProt.server.1.retryTimeOut="500"
voIpProt.server.1.retryMaxCount="3"
voIpProt.SIP.tcpFastFailover="1"
voIpProt.SIP.authOptimizedInFailover="1"
voIpProt.SIP.failoverOn503Response="1"
voIpProt.SIP.outboundProxy.address="sbc.accessSide.com"
voIpProt.SIP.outboundProxy.port="0"
voIpProt.SIP.outboundProxy.transport="TLS"
voIpProt.SIP.outboundProxy.failOver.reRegisterOn="1"
voIpProt.SIP.outboundProxy.failOver.onlySignalWithRegistered="1"
voIpProt.SIP.outboundProxy.failOver.failBack.mode="duration"
voIpProt.SIP.outboundProxy.failOver.failBack.timeout="600"
voIpProt.SIP.outboundProxy.failOver.failRegistrationOn="1"
tcpIpApp.keepalive.tcp.sip.tls.enable="1"
```

```
tcpIpApp.keepalive.tcp.noResponseTransmitInterval="20"
tcpIpApp.keepalive.tcp.idleTransmitInterval="30"
tcpIpApp.keepalive.tcp.sip.persistentConnection.enable="1"
voIpProt.server.X.registerRetry.baseTimeOut="10"
```

# Geo-Redundant SBC's using SRV Records with a Local Survivability Gateway

UCS 5.7 introduces a configuration setting that allows a VVX place the precedence of its static DNS cache above that of a network provided DNS response. This allows an A record controlled by static configuration file an IP address that may be specific to a particular site without impacting devices at other locations who are using the same FQDN for their A record queries.

In this scenario, assume a customer with 1000 branch offices. There are two geographically redundant SBCs that provide access for the phones via the customer office's WAN. Each branch office also has a Local Survivability Gateway (LSG) for routing through the PSTN should the WAN link drop.

The SBCs provide the primary access routes for redundancy services and each require a SIP registration before routing normal SIP traffic. Should one of the SBCs become unavailable, the phone will detect the failure and move to the secondary until failback is triggered. In our example, this is after 3600 seconds has elapsed.

Should the WAN drop or both SBCs become unreachable, each new request by the phone will attempt the SBCs first, in order, before sending an INVITE to the LSG. The delay before reaching the gateway can be tuned using the **`retryTimeout`** and **`retryMaxCount`** settings.

By setting the A record DNS override parameter **`dns.cache.A.networkOverride="1"`**, the phone will first attempt to retrieve A record FQDN to IP mappings from static configuration before trying a network based DNS server. This permits each branch office to use a public SRV record that returns 3 A records, 1 for each SBC and a third "fake" record for the LSG.

The SBC A records will not have local static DNS cache values in the configuration file and so the phone will request IPs from the network. The LSG however, will have a value in the static cache configuration and so a unique IP can be provided in each branch office's configuration file rather than having all phones at each branch receive a common IP based on a globally served DNS A record.

**DNS:**

| SRV Record | Weight | Priority | Port | Targets |
|---|---|---|---|---|
| `_sip._udp.provider.com` | `5` | `10` | `5060` | `sbc01.provider.com`<br>`sbc02.provider.com`<br>`localGateway.provider.com` |

| A Record | IP address |
|---|---|
| `sbc01.provider.com` | `10.10.10.100` |
| `sbc02.provider.com` | `20.20.20.200` |

| UCS Parameter and Value | Notes |
|---|---|
| reg.1.address="555" | The phone's registration AoR |
| voIpProt.SIP.outboundProxy.address="sbc.provider.com" | SBC SRV record, this resolves to 3 A records as above |
| voIpProt.SIP.outboundProxy.port="0" | Set the port to 0 so that SRV requests are used rather than A records for address lookup |
| voIpProt.SIP.outboundProxy.failOver.reRegisterOn="1 | Enables re-registration on failover |
| voIpProt.SIP.outboundProxy.failOver.failBack.mode="duration" | |
| voIpProt.SIP.outboundProxy.failOver.failBack.timeout="3600" | |
| voIpProt.server.1.address="provider.com" | Server Address used in the from field of registrations |
| voIpProt.server.1.retryTimeOut="500" | Control failover timings between routes |
| voIpProt.server.1.retrymaxcount="3" | Control failover timings between routes |
| voIpProt.SIP.optimizedInFailover="1" | Minimize retransmission due to lack of authentication credentials |
| dns.cache.A.networkOverride="1" | Change the preference ordering for receiving DNS A record response to local cache first, and network DNS second. |
| dns.cache.A.1.name="localGateway.provider.com" | A record FQDN |
| dns.cache.A.1.ttl="3600" | How long to cache the static IP address |
| dns.cache.A.1.address="192.168.26.10" | IP address mapped to static the A record |