TECHNICAL REFERENCE

June 2023 | 3725-49225-002A

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.

Poly
345 Encinal Street
Santa Cruz, California
95060

Contents

Introduction **3**

Configuring DNS Query Behaviors **4**

NAPTR, SRV & A Record Usage for an Address 4
Mitigating DNS Failure 5
Dynamic DNS Cache Restore *5*
Static DNS Cache *5*
DNS Ordering Changes While a Phone Is Registered 6
DNSNaptr Records for Mixed Transports Across Multiple Networks 6

Re-Registration on Failover (RROFO) **8**

Triggering Failover 8
When Failover Is Not Triggered 8
Failover Limitations of SIP Response Codes *9*
UDP - Failover Timing Controls 9
TCP/TLS Failover Timing Controls 10
Failover and TLS Require Keep-Alives *10*
UDP & TCP Keep-Alives *11*
Failback 11
Subscriptions and Failover 12
Failover and Emergency Dialing for Shared (SCA) Lines 12
SIP Call-ID Behavior on Failover, Failback, and When Unregistered 13

When All Failover Routes Fail **14**

Controlling Back-off Periods Between Registrations 14

RROFO Configuration **15**

Appendix A: Example Configurations **18**

SIP Using UDP and SBC Failover 18
SIP Using TLS, Keep-Alives, and SBC Failover 18

Introduction

Poly Voice Software (PVOS) and Poly UC Software (UCS) provide a great deal of flexibility in how SIP registration and signaling routes are determined. This Solution Guide provides recommended configuration and context for ensuring your service remains active when faced with network interruption or maintenance shutdowns.

The content in this guide applies to the following models:

- VVX series
- CCX series
- Trio series
- Edge E Series

This solution guide supersedes the following documents:

1. Engineering Advisory 66546 - Configuring Optional Re-Registration on Failover Behavior
2. Technical Bulletin 5844 - SIP Server Fallback Enhancements on Polycom Phones

Terminology

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 outbound proxy/Session Border Controller (SBC) to a different SBC as the result of a detected failure of the original SBC or network. This failure may be transitory in nature or persistent.
- **Failover state** - A SIP registration or "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.
- **Fallback/Fallback Server** - A term used to denote an older form of redundancy where a backup server with limited capabilities is present, typically located on-site, and which may not require a SIP registration. Fallback servers are deprecated in the context of this solution guide in favor of using the recommended Re-Registration on Failover feature provided by Poly phones.
- **Primary SBC/Server** - 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 DNS Query Behaviors

NAPTR, SRV & A Record Usage for an Address

The type of DNS query made by the phone is determined by what information has been configured for a server or outbound proxy.

The default transport protocol value used by Poly phones is `dnsNaptr`, which allows the DNS result set to determine the SIP transport protocol used and which may include several varying transport protocol options of different priorities. If a single transport protocol is used on your service, you may bypass the need to use DNSNaptr records by specifying the desired transport protocol value directly in your phone's configuration using one of `TLS`, `TCPOnly`, or `UDPOnly`.
Once a transport protocol is specified, the phone will use SRV queries to determine the port required for use on that service protocol.

To query an address as an SRV record, there must be no explicit port defined in configuration. The default port value used in PVOS is 0. The below configurations will result in an SRV query being used:

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

In the SRV query fails, PVOS will try the same address again as an A record and assume port 5060.

In the below configuration, because the port of 5060 is explicitly configured, PVOS will not send an SRV query at all and instead make an A record query directly.

```
voIpProt.server.1.address="example.poly.com"
voIpProt.server.1.port="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 may obtain lists of up to four addresses in this manner.

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

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

```
naramata.poly.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.

Mitigating DNS Failure

Poly Voice Software provides two optional features for mitigating DNS failures or network outages.

Dynamic DNS Cache Restore

Dynamic DNS Cache Restore provides a fallback for intermittent DNS outages where a server fails to respond in any way to a query or where the transport layer has provided an indication that the server does not exist or is unreachable. In those circumstances, DNS records that have been previously resolved will continue to use their last known working value beyond the record's Time-To-Live (TTL).

Dynamic DNS Cache Restore *will not* restore a record if a DNS server responds, but that response is a negative response (such as: address not found, or no records provided) or an error response (example: malformed query, request type not implemented, internal DNS server error).

Restoring an expired value is temporary; a configurable TTL determines how long the restored record is used before the next query for that address is sent over the network.

Parameter	Values	Default
dns.cache.dynamicRestore.enable	0 or 1	0
dns.cache.dynamicRestore.ttl	90 to 600, seconds	120

Static DNS Cache

The Static DNS cache can be used to configure address record resolution details for situations where DNS is not available at startup or DNS usage is restricted due to a network policy.

Should a DNS query fail to receive a response, or receive a negative response, the phone will check the configured static cache for a result and use that result for the TTL duration defined. Once the TTL has expired, a network-based query is once again attempted.

For A records, the phone may be further configured to search the static DNS cache before attempting a query over the network.

Parameter	Values	Default
For the below parameters, each record type may have up to 18 cached values, where x = 1 to 18		
dns.cache.A.networkOverride	0 or 1	0
dns.cache.A.x.name	valid hostname string	Null
dns.cache.A.x.ttl	0 to 65535, seconds	300
dns.cache.A.x.address	dotted-decimal IP version 4 address	Null

dns.cache.SRV.x.name	domain name string	Null
dns.cache.SRV.x.ttl	0 to 65535, seconds	300
dns.cache.SRV.x.priority	0 to 65535	0
dns.cache.SRV.x.weight	0 to 65535	0
dns.cache.SRV.x.port	0 to 65535	0
dns.cache.SRV.x.target	domain name string	Null
dns.cache.NAPTR.x.name	domain name string	Null
dns.cache.NAPTR.x.ttl	0 to 65535, seconds	300
dns.cache.NAPTR.x.order	0 to 65535	0
dns.cache.NAPTR.x.preference	string	0
dns.cache.NAPTR.x.flags	string	Null
dns.cache.NAPTR.x.service	string	Null
dns.cache.NAPTR.x.regexp	string	Null
dns.cache.NAPTR.x.replacement	domain name string with SRV prefix	Null

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 received list of records in order of weight/priority.

DNSNaptr Records for Mixed Transports Across Multiple Networks

Available as of UC Software 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 that 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 that 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 phone's route redundancy behavior allowing you to:

- Register to a failover server or SBC before sending other SIP requests
- Ignore SIP signaling from unregistered server or SBCs
- Remain on a failover server or SBC for a set period of time before failback occurs
- Remove a registration with the current route before 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, a **no response** situation 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 that receive 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 an 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 acknowledgment

Failover Limitations of SIP Response Codes

Failover is the result of detecting issues when sending *requests* and not when sending *responses*. When sending responses over UDP where reliable delivery is not guaranteed, the phone cannot know that a response did not arrive, and thus failover cannot be triggered.

Similarly, when responding 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*.

UDP - Failover Timing Controls

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

Parameter	Values	Default
`voIpProt.server.1.retryMaxCount` `reg.1.server.1.retryMaxCount`	`0 to 65535`	`3`
`voIpProt.server.1.retryTimeOut` `reg.1.server.1.retryTimeOut`	`0` to `65535` (milliseconds)	`0`

The settings above allow adjustment of failover speed from route to route until the final route in the list is reached. The final route is repeatedly tried until the complete 32-second transaction period since the original request to the primary SBC has elapsed.

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 defined exponential backoff timers (0.5 s, 1 s, 2 s, 4 s, 4 s, 4 s, etc.) until the complete 32-second transaction timer B/F expiry has been reached. How many of the backoff steps are used depends on the value of the **`retryMaxCount`**.

Example 1 - Poly default settings, RFC 3261 exponential backoff, transport = UDP.

- Initial request attempt (counter == 1)
  - Delay before retransmission if no response: 0.5 seconds
- 1st retransmission (counter == 2)
  - Delay: 1 second
- 2nd retransmission (counter == 3, retries complete)
  - Delay 2 seconds
- Failover to next server

Total delay before failover to the next proxy 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 5 seconds wait after each attempt. (10 seconds total)
- Failover to the 2nd IP. Try twice with 5 seconds wait after each attempt (10 seconds total)
- Failover to the final (3rd) server, retries continue for remaining 32 - 20 = 12 seconds, with intervals defined by RFC 3261 (0.5 s, 1 s, 2 s, 4 s, 4 s, 4 s)

TCP/TLS Failover Timing Controls

As 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 described in RFC 3261 to wait until Timer B or F's 32-second transaction expires. This behavior is overridden by **voIpProt.SIP.tcpFastFailover**, allowing TCP to adhere to a faster timeout.

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 UC Software versions earlier than 6.4, 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.

Parameter	Values	Default
voIpProt.server.1.retryMaxCount reg.1.server.1.retryMaxCount	0 to 65535	3
voIpProt.server.1.retryTimeOut reg.1.server.1.retryTimeOut	0 to 65535 (milliseconds)	0
voIpProt.SIP.tcpFastFailover	0 or 1	0
voIpProt.SIP.tcpFastFailover.timeout	2000 - 5000	5000

Example – TCP using a fastFailover timeout

```
voIpProt.SIP.tcpFastFailover="1"
voIpProt.SIP.tcpFastFailover.timeout="2000"
voIpProt.server.1.retryMaxCount="3" (This is ignored when tcpFastFailover="1")
voIpProt.server.1.retryTimeOut="0"  (Must be set to 0 to use tcpFastFailover)
```

- Time=0, initial request
- Response timeout wait: 2 seconds
- Failover to next server

Failover and TLS Require Keep-Alives

When using TLS, a TCP stream must be maintained for continued communication between the phone and its server. 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 Enable or disable TLS keep-alives	0 or 1	0

tcpIpApp.keepalive.tcp.noResponseTransmitInterval 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 How often a new keep-alive is sent when the phone is idle	10 to 7200	30
tcpIpApp.keepalive.tcp.sip.persistentConnection.enable 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

Typically, 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 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 A configurable payload transmitted in the data portion of the keep-alive.	String, max 255 chars	\r\n\r\n
nat.keepalive.tcp.payload A configurable payload transmitted in the data portion of the keep-alive.	String, max 255 chars	\r\n\r\n \r\n\r\n

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.
By default, an authentication challenge (SIP response code 401 or 407) prompts a new request and so

the phone will retry its primary, which may add a delay. 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 the preferred behavior on detection of a failure is to remain using the failover server for a prolonged duration, see RROFO Configuration to either enable RROFO or to change the failback mode to a setting other than "newRequest".

Subscriptions and Failover

SIP Subscriptions (BLF, BLA, SCA, ACD, etc.) are invalidated once a REGISTER to a new server/SBC is successfully accepted. After Registration, new subscriptions will immediately be triggered on the newly registered server or SBC.

Failover and Emergency Dialing for Shared (SCA) Lines

Use cases where only a shared line is configured on a phone.

All shared calls first attempt to seize a line. If a server/SBC is unavailable, a line seize requests for an emergency call will make several retransmissions up to a maximum of 32 seconds. If all line-seize retransmissions fail, or the timer expires, the emergency call INVITE will be sent regardless of the missing line seize authorization. Decrease the number of line-seize attempts using:

voIpProt.SIP.lineSeize.retries

10 (Default)
allowed values: 3 to 10

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

When an INVITE dialog or other request dialog such as a registration 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. The dialog hasn't yet been terminated and the phone is trying other possible paths to route the request.

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

For a registration, a new call-ID is generated only once the phone has first transitioned to an unregistered state and the dialog terminated. 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 an immediate transition.

See: **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 behaves in the same manner as a failed registration if no additional failover servers had been defined. The randomized delay is designed to alleviate server flooding in cases where power outages or loss of the LAN could result in large numbers of phones attempting to register simultaneously once service is restored.

When the randomized backoff period has elapsed, the phone will attempt to register using 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 Back-off Periods Between Registrations

The algorithm is based on an escalating back off example from RFC 5626.

W = min (max-time, (base-time * (2 ^ consecutive-failures)))

Actual wait time is then randomized between W/2 and W.

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

Example: shorten back off times initially and use increasing delays up to a maximum of 120 seconds between consecutive failures. Assume the following:

```
voIpProt.server.1.registerRetry.maxTimeOut="120"
voIpProt.server.1.registerRetry.baseTimeOut="10
```

Consecutive Failures	0	1	2	3	4	5	6+
Result (W)	**10**	**20**	**40**	**80**	**120**	**120**	**120**
Randomized backoff Min – Max (seconds)	**5-10**	**10 - 20**	**20-40**	**40-80**	**60-120**	**60-120**	**60-120**

RROFO Configuration

Parameter	Values	Default
voIpProt.server.x.failOver.reRegisterOn voIpProt.SIP.outboundProxy.failOver.reRegisterOn reg.x.server.y.failOver.reRegisterOn reg.x.outboundProxy.failOver.reRegisterOn	0 or 1	0
voIpProt.server.x.failOver.onlySignalWithRegistered voIpProt.SIP.outboundProxy.failOver.onlySignalWithRegistered reg.x.server.y.failOver.onlySignalWithRegistered reg.x.outboundProxy.failOver.onlySignalWithRegistered	0 or 1	1
voIpProt.server.x.failOver.failRegistrationOn voIpProt.SIP.outboundProxy.failOver.failRegistrationOn reg.x.server.y.failOver.failRegistrationOn reg.x.outboundProxy.failOver.failRegistrationOn	0 or 1	1
voIpProt.server.x.failOver.failBack.mode voIpProt.SIP.outboundProxy.failOver.failBack.mode reg.x.server.y.failOver.failBack.mode reg.x.outboundProxy.failOver.failBack.mode	duration newRequests DNSTTL registration	duration
voIpProt.server.x.failOver.failBack.timeout voIpProt.SIP.outboundProxy.failOver.failBack.timeout reg.x.server.y.failOver.failBack.timeout reg.x.outboundProxy.failOver.failBack.timeout	0, 60-65535	3600
voIpProt.server.x.failOver.unRegisterOnFailBack voIpProt.SIP.outboundProxy.failOver.unRegisterOnFailBack reg.x.server.y.failOver.unRegisterOnFailBack 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 following 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 an 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 an 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

Duration - The phone tries the primary SBC again after the time specified by `failBack.timeout` expires.

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

(DEPRECATED) 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.

failBack.timeout

The time to wait (in seconds) before failback occurs.

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

Available as of UCS 6.3.0.

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

0 - The registration is terminated silently without sending a REGISTER on the currently registered route.

failOver.enhanced.enable

Available as of UCS 6.4 and later software.
Enhanced Failover applies several changes.

1. When failover has occurred, phones with multiple SIP registrations will failback to a primary as a group. Failback will not be started if any of the SIP lines is in an active call.
2. When DNSNaptr results provide multiple transport protocol options, the phone will try all routes within a transport before moving to a route that uses a different transport protocol.
3. If a large number of routes are available, the inter-route delay is 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

Appendix A: Example Configurations

SIP Using UDP and SBC Failover

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

```
reg.1.address="5551239999"
reg.2.address="5554560000"

voIpProt.server.1.address="server.coreSide.com"
voIpProt.server.1.retryTimeOut="500"
voIpProt.server.1.retryMaxCount="3"

voIpProt.SIP.authOptimizedInFailover="1"
voIpProt.SIP.failoverOn503Response="1"

voIpProt.SIP.outboundProxy.address="sbc.accessSide.com"
voIpProt.SIP.outboundProxy.port="0"
voIpProt.SIP.outboundProxy.transport="UDPOnly"
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"
```

SIP Using TLS, Keep-Alives, and SBC Failover

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

- Failover between SBCs has been set to 4 seconds.
- Enhanced failover is enabled
- Keep-alives are enabled
- Randomized back-off periods before registering again escalate from 5-10 seconds with up to 180 seconds.
- Differences from the UDP example above are illustrated in BLUE TEXT.

```
reg.1.address="5551239999"
reg.2.address="5554560000"

voIpProt.server.1.address="server.coreSide.com"
voIpProt.server.1.retryTimeOut="0"
voIpProt.server.1.registerRetry.baseTimeOut="10"
voIpProt.server.1.registerRetry.maxTimeOut="180"

voIpProt.SIP.tcpFastFailover="1"
voIpProt.SIP.tcpFastFailover.timeout="4000"
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.enhanced.enable="1"
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"
```