Configure SIP Registration

Overview

The Cisco Unified Border Element (CUBE) SIP Registration Proxy feature allows service providers to control the flow of registration messages between a customer's private network and their hosted communications platform.

By controlling routine registration traffic at the customer site, service providers can ensure service availability to local endpoints, while protecting core services from high message loads.

Note H.323 protocol is no longer supported from Cisco IOS XE Bengaluru 17.6.1a onwards. Consider using SIP for multimedia applications.

Feature Information for SIP Registration Proxy

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 1: Feature Information for Support for SIP Registration Proxy on CUBE

Feature Name	Releases	Feature Information
Support for CUBE SIP Registration Proxy	Cisco IOS XE Fuji 16.9.1	CUBE SIP Registration Proxy supports sending outbound registrations from CUBE based on incoming registrations. This feature enables direct registration of SIP endpoints with the SIP registrar in hosted Unified Communications deployments. This feature also provides various benefits for handling CUBE deployments with no IPPBX support. The following commands were introduced or modified: **authentication** (dial peer), **registrar server**, **registration passthrough**, **registration spike**, **show sip-ua registration passthrough status**, **voice-class sip registration passthrough static rate-limit**.

Registration Pass-Through Modes

CUBE uses the following two modes for registration pass-through:

End-to-End Mode

In the end-to-end mode, Cisco Unified Border Element (CUBE) collects the registrar details from the Uniform Resource Identifier (URI) and passes the registration messages to the registrar. The registration information contains the expiry time for rate-limiting, the challenge information from the registrar, and the challenge response from the user.

CUBE also passes the challenge to the user if the register request is challenged by the registrar. The registrar sends the 401 or 407 message to the user requesting for user credentials. This process is known as challenge.

CUBE ignores the local registrar and authentication configuration in the end-to-end mode. It passes the authorization headers to the registrar without the header configuration.

End-to-End Mode--Call Flows

This section explains the following end-to-end pass-through mode call flows:

Register Success Scenario

The successful register scenario for the end-to end registration pass-through mode is as follows:

1. The user sends the register request to CUBE.
2. CUBE matches the request with a dial peer and forwards the request to the registrar.
3. CUBE receives a success response message (200 OK message) from the registrar and forwards the message to the endpoint (user).
4. The registrar details and expiry value are passed to the user.

Registrar Challenging the Register Request Scenario

The following scenario explains how the registrar challenges the register request:

1. The user sends the register request to CUBE.
2. CUBE matches the register request with a dial peer and forwards it to the registrar.
3. The registrar challenges the register request.
4. CUBE passes the registrar response and the challenge request, only if the registrar challenges the request to the user.
5. The user sends the register request and the challenge response to the CUBE.
6. CUBE forwards the response to the registrar.
7. CUBE receives success message (200 OK message) from the registrar and forwards it to the user.

Peer-to-Peer Mode

In the peer-to-peer registration pass-through mode, the outgoing register request uses the registrar details from the local CUBE configuration. CUBE answers the challenges received from the registrar using the configurable authentication information. CUBE can also challenge the incoming register requests and authenticate the requests before forwarding them to the network.

In this mode, CUBE sends a register request to the registrar and also handles register request challenges. That is, if the registration request is challenged by the registrar (registrar sends 401 or 407 message), CUBE forwards the challenge to the user and then passes the challenge response sent by the user to the registrar. In the peer-to-peer mode, CUBE can use the **authentication** command to calculate the authorization header and then challenge the user depending on the configuration.

Note The **registrar** command must be configured in peer-to-peer mode. Otherwise, the register request is rejected with the 503 response message.

Peer-to-Peer Mode--Call Flows

This section explains the following peer-to-peer pass-through mode call flows:

Register Success Scenario

The register success scenario for a peer-to-peer registration pass-through mode is as follows:

1. The user sends the register request to CUBE.
2. CUBE matches the register request with a dial peer and forwards the register request to the registrar.
3. CUBE receives a success message (200 OK message) from the registrar and forwards it to the endpoint (user). The following functions are performed:
    - CUBE picks up the details about the registrar from the configuration.
    - CUBE passes the registrar details and expiry value to the user.

Registrar Challenging the Register Request Scenario

The following scenario explains how the registrar challenges the register request:

1. The user sends the register request to CUBE.
2. CUBE matches the register request with a dial peer and forwards the register request to the registrar.
3. The user responds to the challenge request.
4. CUBE validates the challenge response and forwards the register request to the registrar.
5. CUBE receives a success message from the registrar and forwards it to the endpoint (user).

Registration in Different Registrar Modes

This section explains SIP registration pass-through in the following registrar modes:

Primary-Secondary Mode

In the primary-secondary mode the register message is sent to both the primary and the secondary registrar servers simultaneously.

The register message is processed as follows:

- The first successful response is passed to the phone as a SUCCESS message.
- All challenges to the request are handled by CUBE.
- If the final response received from the primary and the secondary servers is an error response, the error response that arrives later from the primary or the secondary server is passed to the phone.
- If only one registrar is configured, a direct mapping is performed between the primary and the secondary server.
- If no registrar is configured, or if there is a Domain Name System (DNS) failure, the "503 service not available" message is sent to the phone.

DHCP Mode

In the DHCP mode the register message is sent to the registrar server using DHCP.

Multiple Register Mode

In the multiple register mode, you can configure a dial peer to select and enable the indexed registrars. Register messages must be sent only to the specified index registrars.

The response from the registrar is mapped the same way as in the primary-secondary mode.

Registration Overload Protection

The registration overload protection functionality enables CUBE to reject the registration requests that exceed the configured threshold value.

To support the registration overload protection functionality, CUBE maintains a global counter to count all the pending outgoing registrations and prevents the overload of the registration requests as follows:

- The registration count is decremented if the registration transaction is terminated.
- The outgoing registrations are rejected if the count goes beyond a configured threshold.

- The incoming register request is rejected with the 503 response if the outgoing registration is activated by the incoming register request.
- A retry timer set for a random value is used for attempting the registration again if the registrations are originated from CUBE.

The registration overload protection functionality protects the network from the following:

- Avalanche Restart-All the devices in the network restart at the same time.
- Component Failures-Sudden burst of load is routed through the device due to a device failure.

Registration Overload Protection-Call Flow

The following steps explain the register overload protection scenario:

1. The user sends a register request to CUBE.
2. CUBE matches the request with a dial peer and forwards the register request to the registrar.
3. The registration is rejected with a random retry value when the registration threshold value is reached.

Note The call flow for the DNS query on the Out Leg is the same for the end-to-end and peer-to-peer mode.

Registration Rate-limiting

The registration rate-limiting functionality enables you to configure different SIP registration pass-through rate-limiting options. The rate-limiting options include setting the expiry time and the fail count value for a Cisco UBE. You can configure the expiry time to reduce the load on the registrar and the network. CUBE limits the reregistration rate by maintaining two different timers--in-registration timer and out-registration timer.

The initial registration is triggered based on the incoming register request. The expiry value for the outgoing register is selected based on the CUBE configuration. On receiving the 200 OK message (response to the BYE message) from the registrar, a timer is started using the expiry value available in the 200 OK message. The timer value in the 200 OK message is called the out-registration timer. The success response is forwarded to the user. The expiry value is taken from the register request and the timer is started accordingly. This timer is called the in-registration timer. There must be a significant difference between the in-registration timer and the out-registration timer values for effective rate-limiting.

Registration Rate-limiting Success--Call Flow

The following steps explain a scenario where the rate-limiting functionality is successful:

1. The user sends the register request to CUBE.
2. CUBE matches the registration request with a dial peer and forwards it to the registrar. The outgoing register request contains the maximum expiry value if the rate-limiting functionality is configured.
3. The registrar accepts the registration.
4. CUBE forwards the success response with the proposed expiry timer value.

5. The user sends the reregistration requests based on the negotiated value. CUBEresends the register requests until the out-leg expiry timer value is sent.
6. CUBE forwards the subsequent register request to the registrar, if the reregister request is received after the out-leg timer is reached.

Prerequisites

- You must enable the local SIP registrar. See Enable Local SIP Registrar, on page 6 .
- You must configure dial peers manually for call routing and pattern matching

Restrictions

Does not support IPv6.

Configure SIP Registration Proxy

Enable Local SIP Registrar

Perform this task to enable the local SIP registrar.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice service voip**
4. **sip**
5. **registrar server** [**expires** [**max** *value*] [**min** *value*]]
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**voice service voip** **Example:** `Device(config)# voice service voip`	Enters voice-service configuration mode.
Step 4	**sip** **Example:** `Device(conf-voi-serv)# sip`	Enters service SIP configuration mode.
Step 5	**registrar server** [**expires** [**max** *value*] [**min** *value*]] **Example:** `Device(conf-serv-sip)# registrar server`	Enables the local SIP registrar. • Optionally you can configure the expiry time of the registrar using the following keywords: • **expires**--Configures the registration expiry time. • **max**--Configures the maximum registration expiry time. • **min**--Configures the minimum registration expiry time. **Note** The **registrar** command must be configured in peer-to-peer mode. Otherwise, the register request is rejected with the 503 response message.

Step 6	**end**
	Example:
	`Device(conf-serv-sip)# end`
		Exits service SIP configuration mode and returns to privileged EXEC mode.

Configure SIP Registration Proxy at the Global Level

Perform this task to configure SIP registration proxy at the global level.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice service voip**
4. **sip**
5. **registration passthrough** [**system** | [**static** | **dynamic** [**local-fallback** *value*]] [**rate-limit** [**expires** *value*] [**fail-count** *value*]] [**reg-sync** *value*] [**registrar-index** *index*]]
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**voice service voip** **Example:** `Device(config)# voice service voip`	Enters voice-service configuration mode.
Step 4	**sip** **Example:** `Device(conf-voi-serv)# sip`	Enters service SIP configuration mode.
Step 5	**registration passthrough** [**system** | [**static** | **dynamic** [**local-fallback** *value*]] [**rate-limit** [**expires** *value*] [**fail-count** *value*]] [**reg-sync** *value*] [**registrar-index** *index*]] **Example:** `Device(conf-serv-sip)# registration passthrough`	Configures the SIP registration pass-through options. • You can specify different SIP registration pass-through options using the following keywords: • **dynamic**—SIP Registration uses the dynamic registrar details (default). • **local-fallback**—Configures Local Fallback - (e2e). • **rate-limit**—Enables rate-limiting. • **reg-sync**—Sends REGISTER messages when registrar up (p2p). • **registrar-index**—Configures a list of registrars to be used for registration. For detailed information, see Configuring Multiple Registrars on SIP Trunks. • **static**—SIP Registration Use static Registrar Details. • **system**—Use system registration passthrough configuration.
Step 6	**end** **Example:**	Exits service SIP configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(conf-serv-sip)# end`	

Configure SIP Registration Proxy at the Tenant Level

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice class tenant** *tag*
4. **registrar** { **dhcp** | *[registrar index] registrar-server-address [:port]* | **expires** *value*}
5. **registration passthrough** [**system** | [**static** | **dynamic** [**local-fallback** *value*]] [**rate-limit** [**expires** *value*] [**fail-count** *value*]] [**reg-sync** *value*] [**registrar-index** *index*]]
6. **exit**
7. **dial-peer voice** *tag* **voip**
8. **voice-class sip tenant** *tag*
9. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**voice class tenant** *tag* **Example:** `Device(config)# voice class tenant 1`	Enters the tenant configuration mode.

Step 4	**registrar** { **dhcp** | *[registrar index] registrar-server-address [:port]* | **expires** *value*}
	Example:
	`Device(config-class)#registrar ipv4:10.65.75.45:9052 expires 3600`
		Configures the registrar server.

Step 5	**registration passthrough** [**system** | [**static** | **dynamic** [**local-fallback** *value*]] [**rate-limit** [**expires** *value*] [**fail-count** *value*]] [**reg-sync** *value*] [**registrar-index** *index*]]	Configures SIP registration pass-through options on a dial peer on a dial peer.

	Command or Action	Purpose
	Example: `Device(config-class)# registration passthrough static`	- You can specify different SIP registration pass-through options using the following keywords: - **dynamic**—SIP Registration uses the dynamic registrar details (default). - **local-fallback**—Configures Local Fallback - (e2e). - **rate-limit**—Enables rate-limiting. - **reg-sync**—Sends REGISTER messages when registrar up (p2p). - **registrar-index**—Configures a list of registrars to be used for registration. For detailed information, see Configuring Multiple Registrars on SIP Trunks. - **static**—SIP Registration Use static Registrar Details. - **system**—Use system registration passthrough configuration.
Step 6	**exit** **Example:** `Device(config-class)# exit`	Exits tenant configuration mode and returns to global configuration mode.
Step 7	**dial-peer voice** *tag* **voip** **Example:** `Device(config)# dial-peer voice 444 voip`	Enters dial peer voice configuration mode.
Step 8	**voice-class sip tenant** *tag* **Example:** `Device(config-dial-peer)# voice-class sip tenant 1`	Associates the dial-peer with the tenant.
Step 9	**exit** **Example:** `Device(config-class)# exit`	Exits dial-peer configuration mode and returns to global configuration mode.

Configure SIP Registration Proxy at the Dial Peer Level

Perform this task to configure SIP registration proxy at the dial peer level.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *tag* **voip**
4. **voice-class sip registration passthrough** [**system** | [**static** | **dynamic** [**local-fallback** *value*]] [**rate-limit** [**expires** *value*] [**fail-count** *value*]] [**reg-sync** *value*] [**registrar-index** *index*]]
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**dial-peer voice** *tag* **voip** **Example:** `Device(config)# dial-peer voice 444 voip`	Enters dial peer voice configuration mode.
Step 4	**voice-class sip registration passthrough** [**system** | [**static** | **dynamic** [**local-fallback** *value*]] [**rate-limit** [**expires** *value*] [**fail-count** *value*]] [**reg-sync** *value*] [**registrar-index** *index*]] **Example:** `Device(config-dial-peer)# voice-class sip registration passthrough static`	Configures SIP registration pass-through options on a dial peer on a dial peer. • You can specify different SIP registration pass-through options using the following keywords: • **dynamic**—SIP Registration uses the dynamic registrar details (default). • **local-fallback**—Configures Local Fallback - (e2e). • **rate-limit**—Enables rate-limiting. • **reg-sync**—Sends REGISTER messages when registrar up (p2p). • **registrar-index**—Configures a list of registrars to be used for registration. For detailed information, see Configuring Multiple Registrars on SIP Trunks. • **static**—SIP Registration Use static Registrar Details.

	Command or Action	Purpose
		• **system**—Use system registration passthrough configuration.
Step 5	**exit** **Example:** `Device(config-dial-peer)# exit`	Exits dial peer voice configuration mode and returns to global configuration mode.

Configure Registration Overload Protection

Perform this task to configure registration overload protection functionality on CUBE.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **sip-ua**
4. **registration spike** *max-number*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**sip-ua** **Example:** `Device(config)# sip-ua`	Enters SIP user-agent configuration mode.
Step 4	**registration spike** *max-number* **Example:** `Device(config-sip-ua)# registration spike 100`	Configures registration overload protection functionality on CUBE.
Step 5	**end** **Example:**	Exits SIP user-agent configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-sip-ua)# end`	

Configure CUBE to Route a Call to the Registrar Endpoint

Perform this task to configure CUBE to route a call to the registrar endpoint.

Note You must perform this configuration on a dial peer that is pointing towards the endpoint.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *tag* {**pots** | **voatm** | **vofr** | **voip**}
4. **session target registrar**
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 3	**dial-peer voice** *tag* {**pots** | **voatm** | **vofr** | **voip**}
	Example:
	`Device(config)# dial-peer voice 444 voip`
		Enters dial peer voice configuration mode.

Step 4	**session target registrar** **Example:** `Device(config-dial-peer)# session target registrar`	Configures CUBE to route the call to the registrar endpoint.
Step 5	**exit** **Example:** `Device(config-dial-peer)# exit`	Exits dial peer voice configuration mode and returns to global configuration mode.

Verify the SIP Registration

Perform this task to verify the configuration for SIP registration on CUBE. The **show** commands need not be entered in any specific order.

SUMMARY STEPS

1. **enable**
2. **show sip-ua registration passthrough status**
3. **show sip-ua registration passthrough status detail**

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode.

Example:

```
Device> enable
```

Step 2 **show sip-ua registration passthrough status**

Displays the SIP user agent (UA) registration pass-through status information.

Example:

```
Device# show sip-ua registration passthrough status

CallId       Line         peer         mode In-Exp       reg-I Out-Exp
============ ============ ============ ==== ============ ===== ============
771          5500550055   1             p2p  64           1      64
=============================================================================
```

Step 3 **show sip-ua registration passthrough status detail**

Displays the SIP UA registration pass-through status information in detail.

Example:

```
Device# show sip-ua registration passthrough status detail
============================================================
Configured Reg Spike Value: 0
Number of Pending Registrations: 0
============================================================
Call-Id: 763
Registering Number: 5500550055
Dial-peer tag: 601
Pass-through Mode: p2p
Negotiated In-Expires: 64 Seconds
Next In-Register Due in: 59 Seconds
In-Register Contact: 9.45.36.5
----------------------------------------
 Registrar Index: 1
 Registrar URL: ipv4:9.45.36.4
 Negotiated Out-Expires: 64 Seconds
 Next Out-Register After: 0 Seconds
============================================================
```

The following section will be added to the "Examples" section of the SIP to SIP chapter.

Configuration Example-Hosted and Cloud Services SIP Registration Proxy

```
!
!
voice service voip
sip
  registrar server expires max 121 min 61
  registration passthrough static rate-limit expires 9000 fail-count 5 registrar-index 1 3
 5
!
dial-peer voice 1111 voip
 destination-pattern 1234
 voice-class sip pass-thru content unsupp
 session protocol sipv2
 session target registrar
!
dial-peer voice 1111 voip
 destination-pattern 1234
 voice-class sip pass-thru content unsupp
 voice-class sip registration passthrough static rate-limit expires 9000 fail-count 5
registrar-index 1 3 5
 authentication username 1234 password 7 075E731F1A realm cisco.com
 session protocol sipv2
 session target registrar
!
sip-ua
 registration spike 1000
!
!
```