Implementing Cisco Unified Border Element (SP Edition) Policies

A Cisco Unified Border Element (SP Edition) policy is a set of rules that define how the Cisco Unified Border Element (SP Edition) treats different kinds of voice over IP (VoIP) events. A Cisco Unified Border Element (SP Edition) policy allows you to control the VoIP signaling and media that passes through the Cisco Unified Border Element (SP Edition) at an application level.

Note From Cisco IOS XE Release 2.4, configuration of policies is supported in the unified model. Enhancements to this feature have been introduced in later releases.

Cisco Unified Border Element (SP Edition) was formerly known as Integrated Session Border Controller and may be commonly referred to in this document as the session border controller (SBC).

For a complete description of the commands used in this chapter, refer to *Cisco Unified Border Element (SP Edition) Command Reference: Unified Model* at:

http://www.cisco.com/en/US/docs/ios/sbc/command/reference/sbcu_book.html

For information about all Cisco IOS commands, use the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or a Cisco IOS master commands list.

Feature History for Implementing Cisco Unified Border Element (SP Edition) Policies

Release	Modification
Cisco IOS XE Release 2.4	This feature was introduced on the Cisco IOS XR along with support for the unified model.
Cisco IOS XE Release 2.5	Subscriber Policy support, Regular expression based routing support, SIP trunk-group ID routing support, and the SIP media line removal feature were added on the Cisco ASR 1000 Series Routers. Support for H.323 call routing features: H.323 Hunting and multiARQ hunting, Picking a next Hop in Routing Policy, Support for H.323 Addressing, DNS Name Resolution, Number Validation and Editing, Load Balancing, and Inter-VPN Calling were added on the Cisco ASR 1000 Series Routers.
Cisco IOS XE Release 2.6	(Source) Number Analysis feature updated to include source number table and source prefix table.
Cisco IOS XE Release 3.1S	Support for Asymmetric payload types was added.

Cisco IOS XE Release 3.2S	The Number Analysis feature was updated to include source address manipulation. The following number analysis tables were changed: • na-src-number-table was changed to na-src-address-table. • na-dst-number-table was changed to na-dst-address-table. • na-dst-number-attr-table was changed to na-carrier-id-table. • first-number-analysis-table was renamed as first-inbound-na-table Also, first-outbound-na-table was introduced, active-call-policy-set was renamed as call-policy-set default, and active-cac-policy-set was renamed as cac-policy-set global. Administrative domains were introduced. The copy and swap procedure for Call Admission Control (CAC) and call policy sets was introduced. The Multiple CAC Averaging Period feature was added. The Privacy Service feature was added. The Multiple SBC Media Bypass feature was added.
Cisco IOS XE Release 3.3S	Message, Policy, and Subscriber Statistics enhancements were added.
Cisco IOS XE Release 3.4S	The Limiting Resource Usage feature was added.
Cisco IOS XE Release 3.5S	CAC-related enhancements were introduced. The **branch** command has been introduced as an alternative to the **caller** and **callee** command pair in some configuration scenarios.
Cisco IOS XE Release 3.6S	The Common IP Address Media Bypass feature was added.

Contents

This chapter contains the following sections:

Prerequisites for Implementing Policies

The following prerequisites are required to implement Cisco Unified Border Element (SP Edition) policies:

Before implementing policies, Cisco Unified Border Element (SP Edition) must already be configured.

Restrictions

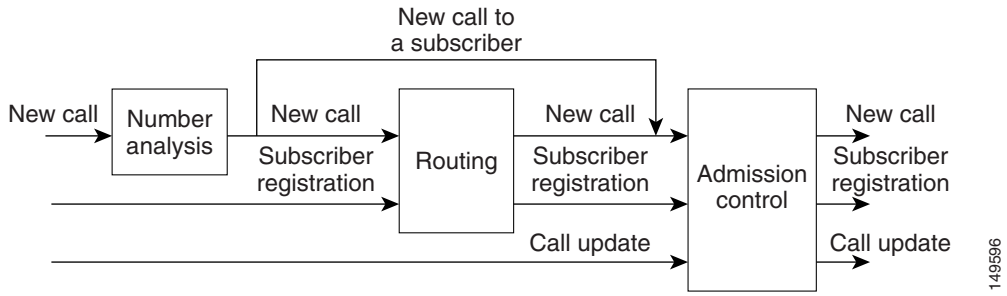
The following restrictions apply when you implement routing policies on the Cisco Unified Border Element (SP Edition):

- H.323 protocols are not supported in Cisco IOS XE Release 2.4 and earlier.
- Regular expression matching is only supported for text user names and domain names in source or destination URIs for SIP calls. Regular expression matching for telephone numbers used in H.323 calls is not supported.
- SBC does not allow addition, modification, or removal of trunk-group ID (TGID) information before call routing occurs.
- SBC does not allow regular expression matching when performing TGID routing.

Information About Implementing Policies

A policy is a set of rules that define how the Cisco Unified Border Element (SP Edition) treats different kinds of VoIP events. A Cisco Unified Border Element (SP Edition) policy allows you to control the VoIP signaling and media that passes through Cisco Unified Border Element (SP Edition) at an application level. Figure 7-1 shows an overview of policy control flow.

Figure 7-1 Policy Control Overview



Number analysis and routing are configured in one type of configuration set, admission control is configured in another.

Number analysis (NA) determines whether a set of source digits or dialed digits represents a valid telephone number (based on number validation, number categorization, or digit manipulation). Call routing determines the VoIP signaling entity to which a signaling request should be sent. A destination adjacency is chosen for the signaling message based on various attributes of the message (for example, based on source account or adjacency). Routing policy is applied to new call events and to subscriber registration events.

In releases earlier than Cisco IOS XE Release 3.2S, textual usernames would bypass NA and proceed to route analysis, where they could be matched. From Cisco IOS XE Release 3.2S, NA can validate both dialed digits and textual usernames.

Also, in releases earlier than Cisco IOS XE Release 3.2S, dst-address in NA could be edited, but not src-address. From Cisco IOS XE Release 3.2S, src-address in NA can also be edited. The task of editing src-address can only be performed on digit strings, as in the case of editing dst-address.

In Cisco IOS XE Release 3.2S, **na-src-name-anonymous-table** command was introduced to determine whether the source number's display name or presentation number is anonymous.

Call Admission Control (CAC) limits the number of concurrent calls and registrations, and restricts the media bandwidth dedicated to active calls. It allows for load control on other network elements by rate limiting. Certain events can be completely blocked (using a blacklist) or freely allowed (using a whitelist), based on certain attributes.

Not all policies are mandatory:

- To call between subscribers, only endpoint routing policy is required.
- To call between telephone numbers, only call routing policy is required.
- Number analysis and admission control are optional, although they are likely to be required by the user.

Policies refer to accounts and adjacencies by name. Therefore, you may find it useful to configure and name adjacencies before configuring policies although this is not required.

The following sections describe the many concepts critical to understanding how to implement Cisco Unified Border Element (SP Edition) policies:

Cisco Unified Border Element (SP Edition) Policies

This section describes the following Cisco Unified Border Element (SP Edition) policies:

Policy Events

Policies are applied to the following events:

- New calls—When new SIP or H.323 calls are signaled to the Cisco Unified Border Element (SP Edition), Cisco Unified Border Element (SP Edition) applies a policy to determine what happens to the new call request and what constraints the call must satisfy during its lifetime.
- Call updates—If one of the endpoints in a call attempts to renegotiate new media parameters, Cisco Unified Border Element (SP Edition) applies policy to ratify the attempt.
- Subscriber registrations—If a subscriber attempts to register through Cisco Unified Border Element (SP Edition), Cisco Unified Border Element (SP Edition) applies policy to determine what happens to the registration request.

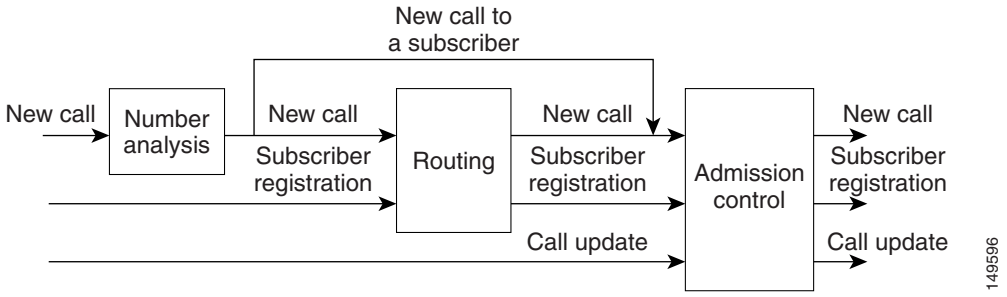
Policy Stages

In the context of SIP and H.323 calls, three distinct stages of a policy are applied in a sequence to the policy events. The stages are:

- Inbound number analysis
- Routing
- Outbound number analysis
- Admission control

Some of these policy stages are skipped for particular types of events. Figure 7-2 shows the sequence of the policy stages for each event type.

Figure 7-2 Policy Stages for Event Types



If the policy stages fail, the call is rejected and the failure is propagated back to the calling device (using either session initiation protocol (SIP) or H.323 signaling, as appropriate) with the error codes in Table 7-2.

Component	Resulting SIP Error Code	Resulting H.323 Error
Number analysis	604 “Does not exist anywhere”	ITU-T Q.931 Release Complete UUIE with H.225 Reason field unreachableDestination
Routing	604 “Does not exist anywhere”	ITU-T Q.931 Release Complete UUIE with H.225 Reason field unreachableDestination
Call Admission Control	503 “Service Unavailable”	ITU-T Q.931 Release Complete UUIE with H.225 Reason field noPermission

Note If the call fails at the routing or Call Admission Control phase, it is released. There is no attempt to retry. Whether or not to retry is left to the upstream (calling) device to decide.

The following sections describe policy stages in more detail:

- Number Analysis
- Routing
- Admission Control

Number Analysis

Number Analysis (NA) determines whether a set of dialed digits or source number represents a valid telephone number. This is achieved by configuring one or more tables of valid source number and dialed digit strings using a limited-form regular-expression syntax, then matching the actual source number or dialed digits against the different strings in the tables.

NA policy is applied only to new call events. If NA determines that a new call does not contain a valid set of source numbers or dialed digits, Cisco Unified Border Element (SP Edition) rejects the call, using the error code described in the ?$paranum>Policy Stages? section.

NA rules are sensitive to the source account and source adjacency of a call, which allows different dial plans to be configured for different customer organizations, or even for different endpoints.

In addition to validating a source number and dialed number, NA policy can also:

- Reformat the dialed digits into canonical form; for example, E.164 format.
- Label the call with a category, which is used by the later stages of policy.

Routing

Routing determines the next-hop VoIP signaling entity to which a signaling request should be sent. Routing of VoIP signaling messages occurs in two stages:

- Policy-based routing—The first stage of routing. In policy-based routing, a destination adjacency is chosen for the signaling message, based on various attributes of the message, discussed later.
- Protocol-based routing—Takes place after policy-based routing. Protocol-based routing uses a VoIP protocol-specific mechanism to deduce a next-hop IP address from the signaling peer configured for the destination adjacency chosen by policy-based routing.

  For example, if the destination adjacency is a SIP adjacency and the signaling peer is uk.globalisp.com, Cisco Unified Border Element (SP Edition) uses domain name server (DNS) or IP lookup to determine the IP address and port of the SIP server for the domain uk.globalisp.com, and forwards the appropriate signaling message to that IP address and port.

Routing policy is applied to new call events and to subscriber registration events.

If a new call event matches an existing subscription, the call is routed automatically to the source IP address and port of the original subscriber registration. No configured policy is required to achieve this, and no configured policy can influence the routing of such calls.

Routing policy is not applied to call update events; call update signaling messages are routed automatically to the destination adjacency that was chosen for the new call event that originated the call.

It is possible that an event cannot be routed, if its attributes do not match a suitable configured routing rule. In such cases, Cisco Unified Border Element (SP Edition) rejects the event using a suitable error code.

Regular expression based routing feature allows the user to configure routing rules that use regular expressions to match the user name or domain part of a source or destination SIP URI.

SBC supports SIP trunk-group ID routing which provides call routing based on the value of the source or destination TGID parameters in the received SIP INVITE message.

Note A trunk in a network is a communication path connecting two switching systems used in the establishment of an end-to-end connection. A trunk-group is a set of trunks, traffic engineered as a unit, for the establishment of connections within or between switching systems in which all of the paths are interchangeable. TGID is a string that identifies a trunk-group uniquely within a given context.

Admission Control

Call admission control determines whether an event should be granted or refused based on configured limits for network resource utilization. There are two reasons for performing admission control.

- To defend load-sensitive network elements, such as softswitches, against potentially harmful levels of load precipitated by singular events, such as DoS attacks, natural or man-made disasters, or mass-media phone-ins.
- To police the Service Level Agreements (SLAs) between organizations, to ensure that the levels of network utilization defined in the SLA are not exceeded.

Call admission control policy is applied to all event types. If an event is not granted by admission control policy, then Cisco Unified Border Element (SP Edition) rejects it with a suitable error code.

Policy Sets

A policy set is a group of policies that can be active on Cisco Unified Border Element (SP Edition) at any one time. If a policy set is active, then Cisco Unified Border Element (SP Edition) uses the rules defined within it to apply policy to events. You can create multiple policy sets on a single Cisco Unified Border Element (SP Edition).

A policy set has two potential uses:

- It enables you to atomically modify the configured policy by creating a copy of the currently active policy set, making all necessary changes, reviewing the modified policy, and then switching the active policy set. If a problem is discovered with the new policy set after it is activated, Cisco Unified Border Element (SP Edition) can be switched back to using the previous policy set with a single command.
- It enables you to create different policy sets for use at different times and to switch between them at the appropriate times.

Number analysis and routing are configured in a call policy set. Admission control is configured in a CAC policy set.

A new policy set can either be created empty (that is, without any configured policies), or created as a copy of another policy set. A policy set can be deleted, provided that it is not the active policy set.

When the Cisco Unified Border Element (SP Edition) is initialized, there are no active policy sets. At any time after initialization, the active policy set can be undefined. While there is no active routing policy, each event that requires routing is rejected.

From Cisco IOS XE Release 3.2S, the administrative domain allows a user to create separate groups of start indexes for number analysis, route analysis, and a CAC policy that can point to different policy sets. The administrative domain is then attached to the adjacencies for both incoming and outgoing analysis stages.

You can designate an inactive call policy set as the active call policy set at any time. However, you cannot directly modify an active call policy set. To modify an active call policy set, perform the copy-and-swap procedure.

You can designate an inactive CAC policy set as the active CAC policy set at any time. You can also modify an active CAC policy set by adding a new table in the CAC policy set. Note that you can create an entry in an existing table of an active CAC policy set only if the table type is **limit all** or **policy-set**. To perform a modification of this type, you must perform the copy-and-swap procedure.

You can define multiple policy sets that are active and select policy sets that can be used at each call analysis stage based on the adjacency setting. To modify a policy that may be referenced by multiple administrative domains, perform the copy-and-swap procedure.

Modifying Active CAC Policy Sets

The procedure to modify an active CAC policy set is the same as the procedure to create a CAC policy set. This procedure is described in the ?$paranum>Configuring Call Admission Control Policy Sets, CAC Tables, and Global CAC Policy Sets? section on page 7-115. The difference lies in the checks the system performs at the end of each of these procedures. The newly modified CAC policy set is activated only after it is determined that the following conditions are met by all the CAC policy tables that are reachable from the modified CAC policy table. A failure message is displayed if any of the CAC policy tables do not meet any of these conditions.

- The table is active.
- All table lookup actions in the table point to valid tables.
- None of the table lookup actions result in a CAC configuration loop.
- All table entry values are valid. For example, the scope name or match prefix length must meet the specified criteria.

Note that the modified CAC policy set is applied only to new incoming calls. Calls that were in progress before the modified CAC policy set is made active are not affected when the modified CAC policy set is made active.

Copy-and-Swap Procedure

To perform a copy and swap procedure, specify the source policy to be copied, and the destination policy to which the source policy is to be copied. The source policy must be an existing policy set, but the destination policy must not be an existing policy set. To protect policies from being overwritten, an error is generated if an attempt is made to copy to an existing policy set.

The old policy can be referenced by different administrative domains, and have multiple indexes within one administrative domain. When the policies are swapped, all the references pertaining to the source policy are replaced with the destination policy. The swap function replaces the default policy and global policy sets, including any policy set referenced in an administrative domain.

The new policy should be set to complete using the **complete** command before all the references to the old policy are replaced.

We recommend that the new policy is exercised globally before all the references to the old policy are replaced.

Note An error is generated if the old policy either does not exist or is in an incomplete state.

The following configuration example describes the steps involved in copying and swapping call policy set 2:

```
Router# show run | b call-policy-set 2
call-policy-set 2
    description this is call policy 1
```

```
    first-call-routing-table TAB1
    first-reg-routing-table TAB2
    rtg-src-adjacency-table TAB1
     entry 1
      match-adjacency SIPP1A
      dst-adjacency SIPP1B
      action complete
     entry 2
      match-adjacency SIPP1B
      dst-adjacency SIPP1A
      action complete
    rtg-src-adjacency-table TAB2
     entry 1
      match-adjacency SIPP1A
      dst-adjacency Registrar
      action complete
     entry 2
      match-adjacency SIPP1B
      dst-adjacency Registrar
      action complete
    complete
```

Step 1 Copy the existing call-policy-set 2 to a new call-policy-set 20:

```
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbc MySBC
Router(config-sbc)# sbe
Router(config-sbc-sbe)# call-policy-set copy source 2 destination 20
```

Step 2 Modify the new call-policy-set with the necessary changes:

```
Router(config-sbc-sbe-rtgpolicy)# first-inbound-na-table InTable
Router(config-sbc-sbe-rtgpolicy)# first-outbound-na-table OutTable
```

Step 3 Set the new call-policy-set 20 to complete:

```
Router(config-sbc-sbe-rtgpolicy)# complete
Router(config-sbc-sbe-rtgpolicy)# exit
```

Step 4 Swap the policies so that references to policy set 2 are replaced with policy set 20. The swap function replaces the default and global policy sets, including any policy set referenced in an administrative domain:

```
Router(config-sbc-sbe)# call-policy-set swap source 2 destination 20
```

The following configuration example describes the steps involved in copying and swapping an existing CAC policy set 12:

```
Router# show run | b cac-policy-set 12
cac-policy-set 12
    first-cac-table 1
    cac-table 1
     table-type limit adjacency
     entry 2
      match-value SIPP1B
      media police strip
      action cac-complete
    complete
```

Step 1 Copy the existing cac-policy-set 12 to a new cac-policy-set 22:

```
Router# configure terminal
```

```
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbc MySBC
Router(config-sbc)# sbe
Router(config-sbc-sbe)# cac-policy-set copy source 12 destination 22
```

Step 2 Modify the new cac-policy-set with the necessary changes:

```
Router(config-sbc-sbe-cacpolicy)# cac-table TAB1
Router(config-sbc-sbe-cacpolicy-cactable)# entry 1
Router(config-sbc-sbe-cacpolicy-cactable-entry)# $max-call-rate-per-scope 100
```

Step 3 Set the new cac-policy-set 22 to complete:

```
Router(config-sbc-sbe)# cac-policy-set 22
Router(config-sbc-sbe-cacpolicy)# complete
Router(config-sbc-sbe-cacpolicy)# exit
```

Step 4 Swap the policies so that references to policy set 12 are replaced with policy set 22:

```
Router(config-sbc-sbe)# cac-policy-set swap source 12 destination 22
```

Policy Tables

All policies on the SBE is configured in a set of tables. This section describes the overall structure of the policy tables, as described in the following sections:

- Nomenclature
- Application of Policy
- Policy Selection
- Policy Table Example

Nomenclature

This section defines some terms that we later use when discussing policy tables.

A policy table has the following properties:

- A name that uniquely identifies the table within the scope of a single policy set. Tables in different policy sets may have the same name.
- A type, which defines the criterion that is used to select an entry from the table.
- A collection of table entries.

A policy table entry is a member of a policy table. It has the following properties:

- A value to match on (the match value). The semantics of this value are determined by the table type. No two entries in the same table may have identical match values.
- An optional *action* to perform on the event, if it matches this entry.
- An optional name of the next table to search for policy, if the event matches this entry.

Application of Policy

The policy tables are searched whenever an event occurs. The policy to be applied to the event is built up as the tables are searched.

The policy sets contains the following properties, which define which policy tables are searched at each stage of the policy calculation. The call policy set contains:

- First NA policy table to process

- First routing policy table to process for calls
- First routing policy table to process for endpoint registrations

The CAC policy set contains the first admission control policy table.

When an event occurs, the policy tables are searched as follows. This procedure is followed once for every stage of policy to which an event is subjected.

- The first table for the particular stage of the policy calculation is obtained from the active configuration set.
- The type of the table defines which of the event's attributes (for example, the destination number or the source adjacency) is being examined by this table.
- This attribute is compared against the match value of every entry in the table. This results in either exactly one entry matching the event, or no entries matching the event.
- If an entry matches the event, then the action associated with that entry is performed. After the action is performed, if the entry contains the name of a next table, that table is processed. If there is no next table, then the policy calculation is complete and processing for this stage of policy ends.
- If no entry matches the event, then the policy calculation is complete and processing for this stage of policy ends.

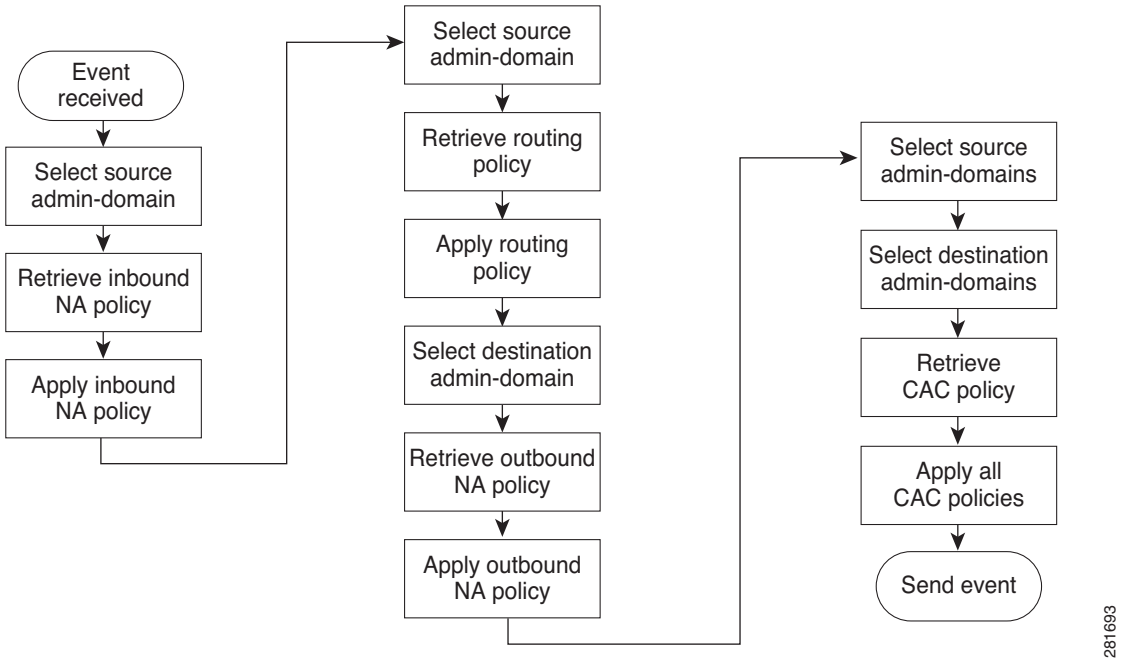
Policy Selection

From Cisco IOS Release 3.2S, the SBC can have multiple active configuration sets. However, by using administrative domains, you can select different policy sets for inbound number analysis, routing, CAC, and outbound number analysis for messages based on their source and destination adjacencies. Figure 7-3 explains the call processing flow using the policy sets.

The policy set that is to be used for a given administrative domain is defined in the admin-domain mode. Call policy sets specified in the admin-domain mode is given a priority. The priority is required because more than one administrative domain can be specified on an adjacency. The SBC will use the policy-set with the highest priority.

The policy sets must be in a complete state before they are assigned to an administrative domain. A default call-policy-set must be configured before the administrative domain mode is entered. If an inbound NA set, a routing set, or an outbound NA set is undefined, the administrative domain uses the values defined within the default call-policy-set. For more information on administrative domains, see the Administrative Domains, page 7-59 section.

Figure 7-3 Call Processing Flow Using Policy Sets



Call Policy

A signaling event is assigned to the default call policy set if an admin-domain is not specified on the adjacency.

However, you can use different sets of incoming and outgoing number analysis tables based on the administrative domains configured for the incoming and outgoing adjacencies respectively. You can also configure a different routing policy set on a per-adjacency basis.

If more than one administrative domain is associated with the incoming adjacency, the SBC will use the policy set with the highest priority. You should not configure two routing policy sets with the same priority, two inbound NA policy sets with the same priority, or any two outbound NA policy sets with the same priority. The SBC logs an error but uses the policy with the highest index value.

If the adjacencies list any administrative domains that is not listed in the admin-domain mode, they use the priority in the global policy. The SBC logs a configuration warning if an adjacency references an undefined administrative domain.

CAC Policy

All events are limited by the applicable CAC policies indicated by the source and destination administrative domains and the global CAC policy.

The user can configure a CAC policy using different sets of tables based on the administrative domains configured on both the incoming and outgoing adjacencies. It is not required by the administrative domain to specify a CAC policy-set.

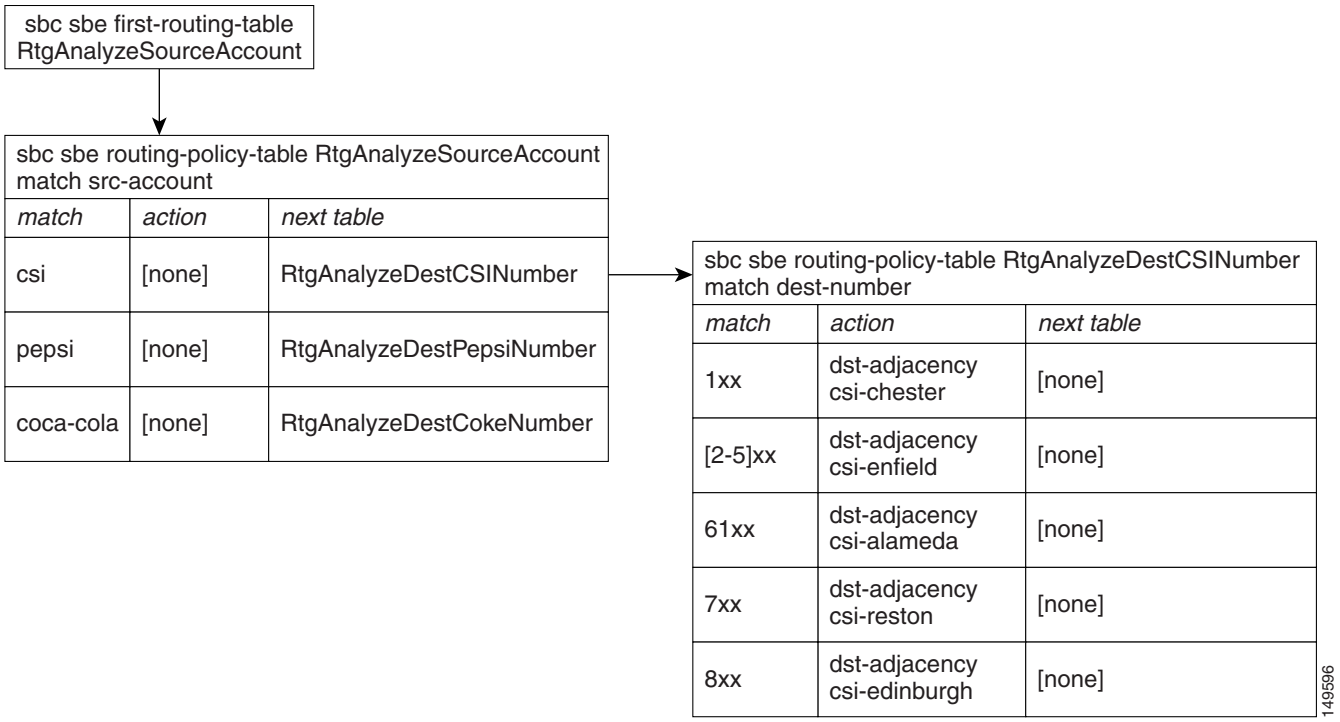
Policy Table Example

The following example illustrates the flow of control as policy tables are parsed at a particular stage of policy for a particular event. The event in this example is a new call, received from source account with destination number **129**. The stage of policy considered here is **routing**.

This example is provided for illustrative purposes only; routing tables are described in detail in the ?$paranum>Routing? section.

Figure 7-4 shows the relevant routing tables.

Figure 7-4 Policy Table Example



The policy calculation begins by looking up the first policy table to be used by the routing stage. This is the table with name RtgAnalyzeSourceAccount. This table is processed as follows:

- The table type of the table is src-account, so the source account of the new call event is compared with each of the entries in this table.
- The table entry that matches on csi provides a match for this new call event. There is no action associated with this entry, but the entry points to a next table with name RtgAnalyzeDestCSINumber.

The flow of control then passes to the table with name RtgAnalyzeDestCSINumber. This table is processed as follows:

- The cac-scope of the table is dst-number, so the destination number of the new call event is compared with each of the entries in this table.
- The table entry that matches on 1xx provides a match for this new call event. The action associated with this entry is performed; that is, the destination adjacency for the new call event is set to csi-chester.
- This entry does not point to a next table, so the policy calculation for the routing stage ends.

This example shows successful routing of the new call. The outcome is successful because the destination adjacency of the new call is selected before the policy calculation finishes. It is entirely possible for the outcome of routing to be unsuccessful for a new call if the routing policy tables do not

assign a destination adjacency to the call before the routing policy calculation ends. For example, the routing policy illustrated above does not successfully route a new call whose source account is csi and whose destination number is 911.

In this example, a single entry is selected from each table that is traversed during the calculation. In general, at most one entry in any policy table matches an event to which policy is being applied. In cases in which more than one entry would match an event, the best matching entry is selected.

Number Analysis Policies

The following Number Analysis (NA) policies are configured within NA tables and are applied simultaneously to new calls and are described in the following sections:

- Number Validation
- Number Categorization
- Digit Manipulation
- Text Addresses
- Outbound Number Analysis

Number Validation

Number validation is fundamental to the process of traversing number analysis policy tables. A number is validated if the NA tables are traversed and the final entry examined contains an action of **accept**. A number is not valid if the NA tables are traversed, and the final entry examined contains an action of **reject**. A number also is not valid if, at any stage of processing the NA tables, a table with no matching entries is encountered.

Number analysis tables can be one of the following types:

- **dst-number**—Tables of this type contain entries whose match values represent complete numbers of Destination. In such tables, an entry matches an event if the entire dialed digit string exactly matches the match value of the entry.
- **dst-prefix**—Tables of this type contain entries whose match values represent number prefixes of Destination. In such tables, an entry matches an event if there exists a subset of the dialed digit string, consisting of consecutive digits taken from the front of the dialed digit string, that exactly matches the match value of the entry.
- **src-number**—Tables of this type contain entries whose match values represent complete numbers of Source. In such tables, an entry matches an event if the entire source digit string exactly matches the match value of the entry.
- **src-prefix**—Tables of this type contain entries whose match values represent number prefixes of Source. In such tables, an entry matches an event if there exists a subset of the source digit string, consisting of consecutive digits taken from the front of the source digit string, that exactly matches the match value of the entry.
- **src-account**—Tables of this type contain entries whose match values are the names of accounts. In such tables, an entry matches an event if the name of the source account of the event exactly matches the match value of the entry.
- **src-adjacency**—Tables of this type contain entries whose match values are the names of adjacencies. In such tables, an entry matches an event if the name of the source adjacency of the event exactly matches the match value of the entry.

- **carrier-id**—Tables of this type contain entries matching the carrier ID.

Digit Matching NA Tables

The format of the match values of entries in NA tables that match on the destination number or destination number prefix is a limited-form, regular expression string representing a string of dialed digits. The syntax used is described in Table 7-1.

Table 7-1 Syntax Used for Digit Matching NA Tables

Syntax Element	Description
X	Any numerical digit 0 – 9.
( )	The digit within the parentheses is optional. For example, **(0)XXXX** represents **0XXXX** and **XXXX**.
[ ]	One of the digits within the square brackets is used. For example, **[01]XXX** represents **0XXX** and **1XXX**. A range of values can be represented within the square brackets. For example, **[013-5]XXX** represents **0XXX**, **1XXX**, **3XXX**, **4XXX** and **5XXX**.
*	The ***** key on the telephone.
#	The **#** key on the telephone.
-	Digit delimiter
,	Digit delimiter
a-f/A-F	Hexadecimal digits

In such tables, it is always possible that more than one entry in the table may match a particular digit string. For example, entries that match 1xx and 12x both match a digit string 129. However, a single entry must be chosen from each table, so the Cisco Unified Border Element (SP Edition) chooses the best matching entry by applying the following rules in the order given.

Step 1 Choose the longest explicit match.

If the NA table is a dst-prefix type, it is possible that more than one entry specifies an explicit number (that is, one that contains no X characters or [ ] constructs) and matches the dialed number of the event. In this situation, the entry with the longest number has priority.

For example, the dialed number begins 011, the number validation table is a dst-prefix type, and there are two matching entries with numbers 01 and 011. The entry with the number 011 takes priority, because it is a longer number.

Step 2 If there is no explicit match, choose the longest wildcard match.

If the table does not contain an explicit entry to match the dialed number of the event, the longest wildcard entry that matches takes priority.

Step 3 If there are multiple wildcard matches of the same length, choose the most explicit where possible.

For example, the dialed number is 01234567890, the NA table is a dst-number type, and there are two matching entries with match values 0123XXXXXXX and 0123456XXXX. In the first entry, the fifth digit is a wildcard; in the second entry, the eighth digit is a wildcard, so the second entry takes priority.

If the same number is dialed, and a different NA table has matching entries [01]234XXXXXXX and 0XXXXXXXXXX, the second entry takes priority, because in the first entry the first digit is a wildcard.

Number Categorization

Events can be placed into user-defined categories during NA processing. This is achieved by specifying a categorization action in an entry of an NA table. Categories are useful, because they may be referred to later during the admission control policy stage.

At most, one category may be associated with an event. If, during processing of the NA tables, categories are assigned to an event multiple times, then the last category to be assigned is used. When a category is assigned to an event, it cannot be deleted, only replaced with another category.

Digit Manipulation

During number analysis (NA), it is often a requirement to normalize numbers—in other words, convert them from the internal format used by a particular organization or service provider to a canonical format understood globally in the Internet and PSTN.

This is achieved by specifying one or more of the following actions in an entry of an NA table:

- *del-prefix N*—This action removes the leading *n* digits from the dialed digit string, or deletes the entire string if it is *n* or fewer digits long.
- *del-suffix n*—This action removes the final *n* digits from the dialed digit string, or deletes the entire string if it is *n* or fewer digits long.
- *add-prefix digit string*—This action adds the given digit string to the front of the dialed digit string.
- *replace digit string*—This action replaces the entire dialed digit string with the given digit string.

Text Addresses

From Cisco IOS XE Release 3.2S, NA supports both textual username and digit matching. The table name na-src-number-table was changed to na-src-address-table, na-dst-number-table to na-dst-address-table, and na-dst-number-attr-table to na-carrier-id-table.

To match the text addresses, the existing match number is modified to read the match address. The **match-address** command can include a suffix of digits or regex.

In number analysis, you can define the following matching criteria types:

- Digit matching matches the dialed digit strings using specialized digit regex.
- Regex matching is applicable only to textual usernames, and offers a basic regular expression (BRE) syntax.

Note Comparison of dialed digits and regex is possible. To compare a fixed string, a regex without any regex metacharacters should be used.

Outbound Number Analysis

Outbound Number Analysis allows the configuration of the source and destination numbers from the canonical form to a form that is appropriate for the destination administrative domains. The configuration of Outbound Number Analysis is similar to that of Inbound Number Analysis, which is converted from the source administrative domain form to the canonical form.

Outbound Number Analysis is performed automatically after successful routing. Outbound Number Analysis is processed using the **call-policy-set outbound-na** command in the destination administrative domain.

Routing

This section describes the following routing policies:

- Routing Tables and Adjacencies
- Number Manipulation
- Hunting
- Regular Expression-Based Routing

Routing Tables and Adjacencies

This section explains how routing tables are configured on the Cisco Unified Border Element (SP Edition).

The inputs to the policy-based routing stage are as follows:

- The destination number of the event, which is the post-NA dialed digit string (that is, it may have been modified from the original dialed digit string)—This input is present only if the event is a new call.
- The source number of the event—This input is present only if the event is a new call.
- The source adjacency of the event.
- The source account of the event.

The routing policy tables examine some or all of these inputs, and produce one of the following outputs:

- A single destination adjacency.
- A group of adjacencies used for load balancing. One of these is chosen, depending on the load previously sent to the adjacencies in this group.

Routing tables represent one of the following types:

- **dst-address**—Tables of this type contain entries matching the dialed number (after number analysis). These values are either complete numbers or number prefixes (depending on whether the *prefix* parameter is given). Without the *prefix* parameter, an entry matches an event if the dialed digit string exactly matches the match value of the entry. With the *prefix* parameter, an entry matches an event if there exists a subset of the dialed digit string, consisting of consecutive digits taken from the front of the dialed digit string that exactly matches the match value of the entry.

  Routing actions also match text user name using a regular expression rather than a literal text string. Routing actions are considered to match if the regular expression matches at least one part of the address.

- **src-address**—Tables of this type contain entries matching the dialer's number or SIP user name. These values are either complete numbers or number prefixes (depending on whether the *prefix* parameter is given). Without the *prefix* parameter, an entry matches an event if the entire digit string representing the calling number exactly matches the match value of the entry. With the *prefix* parameter, an entry matches an event if there exists a subset of the digit string that represents the calling number, consisting of consecutive digits taken from the front of this string that exactly match the match value of the entry.

Routing actions also match text user name using a regular expression rather than a literal text string. Routing actions are considered to match if the regular expression matches at least one part of the address.

- **src-account**—Tables of this type contain entries matching the names of accounts. In such tables, an entry matches an event if the name of the source account of the event exactly matches the match value of the entry.
- **src-adjacency**—Tables of this type contain entries matching the names of adjacencies. In such tables, an entry matches an event if the name of the source adjacency of the event exactly matches the match value of the entry.
- **src-domain**—Tables of this type contain entries matching the source domain names.

  Routing actions also match domain names using full regular expressions rather than the limited range of regular expression matching. Routing actions are considered to match if the regular expression matches at least one part of the domain.
- **dst-domain**—Tables of this type contain entries matching the destination domain names.

  Routing actions also match domain names using full regular expressions rather than the limited range of regular expression matching. Routing actions are considered to match if the regular expression matches at least one part of the domain.
- **carrier-id**—Tables of this type contain entries matching the carrier ID.
- **round-robin-table**—A group of adjacencies are chosen for an event if an entry in a routing table matches that event and points to a round-robin adjacency table in the next-table action. A round-robin adjacency table is a special type of policy table, whose events do not have any match-value parameters, nor next-table actions. Its actions are restricted to setting the destination adjacency and performing digit manipulation.
- **category**—Tables of this type contain entries matching on the category that was assigned to the call during number analysis. You assign the category during number analysis.
- **time**—Tables of this type contain entries matching on a user-configured time. The entries can have overlapping match periods. Time periods can be specified by year, month, date, day of the week, hour, or minute.
- **least-cost**—Tables of this type contain entries matching on the user-configured precedence (cost) of the entries. If more than one entry has an equal cost, an entry is selected based on a user-configured weight or an entry is selected based on the number of active calls on each route. If routing fails, then the adjacency with the next lowest cost is selected.
- **src-trunk-group-id**—Tables of this type contain entries matching the source TGID or TGID context parameters and action type to perform the call routing.
- **dst-trunk-group-id**—Tables of this type contain entries matching the destination TGID or TGID context parameters and action type to perform the call routing.

The rules specified in the ?$paranum>Digit Matching NA Tables? section on page 7-15 govern the format and matching rules of the match-values of the entries in routing tables of type dst-number, dst-prefix, src-number and src-prefix.

Number Manipulation

The number manipulation feature enables you to specify various number manipulations that can be performed on a dialed number after a destination adjacency has been selected. Number manipulation can be configured as a routing policy.

This enhancement affects the billing functionality as it allows the Cisco Unified Border Element (SP Edition) to display both the original and the edited dialed number for a call. For example:

```
<party ty”e="o”ig" pho”e="01234567890”/>
<party ty”e="t”rm" pho”e="23456789”31" editphone=”1111111111111”/>
```

Note The phone numbers in the above example are not real.

The number manipulation feature requires that the edit action be allowed in the routing policy entries. The edit action takes the same parameters as the edit action for the number analysis tables, enabling you to delete a number of characters from the beginning or end of the dialed string, add digits to the start of the string, or replace the entire string with another. For example, if the following table were matched:

```
call-policy-set 1
  rtg-src-adjacency-table table1
    entry 1
      match SipAdj1
      edit del-prefix 3
      dst-adjacency SipAdj2
      action complete
    end
  end
```

then the dialed string would have the first three of its digits deleted.

In the number analysis stage you can specify categories as shown below.

```
call-policy-set 1
    first-inbound-na-table check-accounts
    na-src-account-table check_accounts
     entry 1
      match-account hotel_foo
      action next-table hotel_dialing_plan
     entry 2
      match-account hotel_bar
      action next-table hotel_dialing_plan
     entry 3
      match-account internal
      action accept
    na-dst-prefix-table hotel_dialing_plan
      entry 1
       match-prefix XXX
       category internal
       action accept
     entry 2
      match-prefix 9XXX
      category external
      action accept
```

Later during routing, the calls are routed based on assigned categories.

```
  call-policy-set 1
    first-call-routing-table start_routing
    rtg-category-table start_routing
      entry 1
       match-category internal
       action next-table internal_routing
      entry 2
       match-category external
       action next-table external_routing
    rtg-src-adjacency-table internal_routing
      entry 1
        match-adjacency sip_from_foo
```

```
    dst-adjacency sip_to_foo
    action complete
   entry 2
    match-adjacency sip_from_bar
    dst-adjacency sip_to_bar
    action complete
  rt-dst-address-table external_routing
   entry 1
    match-address 208111
    prefix
    dst-adjacency sip_to_foo
    action complete
   entry 2
    match-address 208222
    prefix
    dst-adjacency sip_to_bar
    action complete
   entry 3
    match-address 208333
    prefix
    dst-adjacency sip_to_softswitch
    action complete
```

Note The category of a call cannot be changed in a routing table. Categories are only assigned during number analysis.

You can also specify various number manipulations to be performed on a dialing or dialed number after a destination adjacency is selected.

The following example adds a prefix of "123" to the source number, for all calls coming in on "SipAdj1" adjacency and destined to "SipAdj2".

```
call-policy-set 1
  rtg-src-adjacency-table table1
    entry 1
      match SipAdj1
      edit-src add-prefix 123
      dst-adjacency SipAdj2
      action complete
```

Hunting

Cisco Unified Border Element (SP Edition) can hunt for other routes or destination adjacencies in case of a failure. Hunting means the route is retried. Cisco Unified Border Element (SP Edition) supports hunting of SIP and H.323 calls. Hunting can be configured as a routing policy.

There are several ways in which failures can occur, including the following:

- CAC policy refusing to admit a call

  If a CAC policy rejects a call, the SBC automatically attempts to reroute the call using the Routing Policy Service (RPS). RPS decides where to route onward signaling requests by using the configured policy in the RPS. The call is then tested against CAC policy again.

- Routing Policy Services being unable to route a call
- Call setup failure being received from SIP or H.323.

  When the SBC receives a call setup failure notification from H.323 or SIP, it is notified whether or not it should attempt to reroute the call, depending upon the error code.

If an SIP or H.323 adjacency attempts to route a call, and the attempt fails, it receives an error code. You can configure which error codes trigger hunting or rerouting.

- If the error code received by the adjacency matches an entry on this list, RPS is signalled to reroute the call. Rerouting then occurs unless the number of attempts exceeds the limit set as the maximum number of routing attempts that SBC makes. The default is three attempts.
- If the error code received by the adjacency does not match an entry on this list, RPS is signalled not to reroute the call.

For both SIP and H.323 call, you can configure a list of error codes or failure return codes to trigger hunting or rerouting for a particular adjacency by using the **sip hunting-trigger error-codes** or **hunting-trigger error-codes** commands.

You can also configure a list of H.323 error codes at a global level, by using the **hunting-trigger** command in the global H.323 configuration mode.

SIP error codes are numeric error codes. H.323 error codes are textual. See the ?$paratext[TC_TableCap,TCW_TableCapW,TCPr_TableCapPref,TCWPr_TableCapWPref,TCF_TableCapPartFirst,TCN_TableCapPartNext,TCWF_TableCapWPartFirst,TCWN_TableCapWPartNext]>? table.

Hunting finishes when one of the following conditions is met:

- The call is successfully routed.
- The SBC receives a call setup failure notification with the instruction not to continue hunting, in which case the call fails.
- The SBC has made the number of specified routing attempts and the call has not been successfully routed, in which case the call fails.
- The SBC has tried all available adjacencies, and the call has not been successfully routed, in which case the call fails.

H.323 hunting has the additional hunting modes of alternate endpoints and multiARQ hunting. See the ?$paranum>H.323 Call Routing Features? section on page 7-24.

For information on configuring SIP and H.323 hunting, see the ?$paranum>Configuring Hunting? section on page 7-107.

Table 7-2 lists the supported error codes that you can configure to trigger hunting of SIP or H.323 calls.

Table 7-2 Configurable Error Codes to Trigger Hunting

Supported SIP Error Codes	Supported H.323 Error Codes
400 - Bad Request	unreachableDestination
401 - Unauthorized	noPermission
402 - Payment Required	noBandwidth
403 - Forbidden	destinationRejection
404 - Not Found	gatewayResources
405 - Method Not Allowed	badFormatAddress
406 - Not Acceptable	securityDenied
407 - Proxy Authentication Required	the internally-defined value “connectFailed”
408 - Request Timeout	—
409 - Conflict	—
410 - Gone	—
411 - Length Required	—
413 - Request Entity Too Large	—
414 - Request URI Too Long	—
415 - Unsupported Media Type	—
416 - Unsupported URI Scheme	—
420 - Bad Extension	—
421 - Extension Required	—
423 - Interval Too Brief	—
480 - Temporarily Unavailable	—
481 - Call/Transaction Does Not Exist	—
482 - Loop Detected	—
483 - Too Many Hops	—
484 - Address Incomplete	—
485 - Ambiguous	—
486 - Busy Here	—
487 - Request Terminated	—
488 - Not Acceptable Here	—
491 - Request Pending	—
493 - Undecipherable	—
500 - Server Internal Error	—
501 - Not Implemented	—
502 - Bad Gateway	—
503 - Service Unavailable	—
504 - Server Time-Out	—

Table 7-2 Configurable Error Codes to Trigger Hunting (continued)

Supported SIP Error Codes	Supported H.323 Error Codes
505 - Version Not Supported	—
513 - Message Too Large	—
600 - Busy Everywhere	—
603 - Declined	—
604 - Does Not Exist Anywhere	—
605 - Not Acceptable	—

Regular Expression-Based Routing

Regular expression based routing allows the user to configure routing rules that use regular expressions to match the user name or domain part of a source or destination SIP URI.

Routing actions match text user name using a regular expression rather than a literal text string when “regex” keyword is used. Routing actions are considered to match if the regular expression matches at least one part of the address.

Table 7-3 shows the basic regular expression (BRE) implementation for the supported regex characters.

Table 7-3 BRE Implementation

Metacharacter	Description
.	Matches any single character. Within POSIX bracket expressions, the dot character matches a literal dot. For example, a.c matches "abc", etc., but [a.c] matches only "a", ".", or "c".
[ ]	A bracket expression. Matches a single character that is contained within the brackets. For example, [abc] matches "a", "b", or "c". [a-z] specifies a range which matches any lowercase letter from "a" to "z". The - character is treated as a literal character if it is the last or the first character within the brackets, or if it is escaped with a backslash: [abc-], [-abc], or [a\-bc].
[^]	Matches a single character that is not contained within the brackets. For example, [^abc] matches any character other than "a", "b", or "c". [^a-z] matches any single character that is not a lowercase letter from "a" to "z". As above, literal characters and ranges can be mixed.
^	Matches the starting position of the string.
$	Matches the ending position of the string.
\(\)	Defines a marked subexpression. The string matched within the parentheses can be recalled later (see the next entry, \n).
\n	Matches what the nth marked subexpression matched, where n is a digit from 1 to 9. This construct is theoretically irregular and was not adopted in the POSIX ERE syntax. Some tools allow referencing more than nine capturing groups.
*	Matches the preceding element zero or more times.
\{m,n\}	Matches the preceding element at least m and not more than n times. For example, a\{3,5\} matches only "aaa", "aaaa", and "aaaaa".

The rtg-src-address and rtg-dst-address tables contain entries matching the dialed number (after number analysis). At run-time, when the Request-URI is processed, the username is parsed to determine if the username is considered to be “textual” or “dialed-digits”. It is initially assumed that the username is a dialed-digit string, and the username will considered to be textual only if non-dialed digit characters are encountered. Having determined this type, only policy entries matching this type are evaluated.

When configuring policy entries which match on rtg-src-address or rtg-dst-address table, it is important to configure the match-address correctly to ensure the policy entry is evaluated. In order to assist in configuration, the type of match address will be assessed and configured automatically if not specifically configured.

You can configure one of the following three choices explicitly:

match-address *address* [**digits**] (limited digit string regex)

match-address *address* [**string**] (string (textual) comparison on textual username only)

match-address *address* [**regex**] (regular expression on string (textual) usernames only)

Example:

Valid entries:

```
match-address (0)1234[56] digits
match-address username string
match-address [Uu]sername regex
```

Invalid entries:

```
match-address 1234 string (cannot perform a string match on dialed digits)
match-address 1234 regex (cannot perform a regex match on dialed-digits)
match-address [abc] regex (abc are valid dialed digits and #, * and d are also valid
dialed digits)
```

In this case the entry is evaluated at configuration time and error responses generated if there is a perceived mismatch in the type and match-address.

H.323 Call Routing Features

In addition to the features described in the ?$paranum>Routing? section on page 7-17 that also apply to H.323 calls, Cisco Unified Border Element (SP Edition) supports various H.323-specific call routing features.

The H.323 call routing features are:

- H.323 Hunting, page 7-25
- Picking a Next Hop in Routing Policy, page 7-26
- Support for H.323 addressing, page 7-26
- DNS Name Resolution, page 7-26
- Number Validation and Editing, page 7-26
- Load Balancing, page 7-27
- Inter-VPN Calling, page 7-27

H.323 Hunting

Cisco Unified Border Element (SP Edition) supports hunting of H.323 calls. Cisco Unified Border Element (SP Edition) hunts for other routes or destination adjacencies in the event of a failure. Hunting re-routes the call in response to a specific user-configured event or error code.

H.323 hunting or re-routing operates in the following ways based on whether the adjacency is a gatekeeper or non-gatekeeper adjacency:

- For a gatekeeper adjacency, the SBC can cycle through a list of potential signaling next hops based on input from the gatekeeper. Alternate Endpoints and MultiARQ are two methods that allow the gatekeeper to provide the SBC with this list.

  If H.323 has a list of alternate endpoints for a call, H.323 tries each of these in turn before reporting a routing failure to the RPS.

  MultiARQ is described in the ?$paranum>MultiARQ Hunting? section.

- For a non-gatekeeper adjacency, or where all the next hops on a gatekeeper adjacency have been exhausted, the SBC can re-route the call to a different adjacency in the “hunt group” (specifically, the round-robin-table or least-cost routing table). For more information on routing tables, see the ?$paranum>Routing Tables and Adjacencies? section on page 7-17.

MultiARQ Hunting

Cisco Unified Border Element (SP Edition) supports a non-standard H.323 mechanism for hunting for other routes or destination adjacencies. This is based on issuing multiple Admission Requests (ARQs) to a Gatekeeper for a single call.

The SBC sends an ARQ (Admission Request) when an incoming call is received on a gatekeeper adjacency, or an outgoing call needs to be made on a gatekeeper adjacency. For an outgoing call, the gatekeeper returns the signaling address of the endpoint that the SBC should contact.

MultiARQ hunting occurs under the following circumstances:

- The H.323 endpoint sends an ARQ to a Gatekeeper as part of establishing an outbound call leg.
- The Gatekeeper contacts other network entities and identifies one or more potential endpoints.
- The Gatekeeper returns an admissionConfirm (ACF) containing a single destinationInfo and no alternateEndpoints.
- The H.323 endpoint attempts to contact the endpoint identified in the ACF. The endpoint either rejects the call or is unreachable.

The MultiARQ hunting continues until one of the following conditions is met.

- An endpoint is contacted and the call completes.
- A Gatekeeper ARQ retry is required, but the hard-coded limit on the number of permitted retry ARQs has been reached. This number is a customizable constant in h323cust.h, and is currently set to 32.
- The Gatekeeper returns an admissionReject, indicating that there are no further suitable endpoint identifiers.
- An endpoint returns a rejectReason which is not configured as a hunting trigger.
- An endpoint cannot be contacted, and connectFailed is not configured as a hunting trigger.

For information on configuring MultiARQ Hunting, see the ?$paranum>Configuring H.323 MultiARQ Hunting? section on page 7-112.

Picking a Next Hop in Routing Policy

When receiving an incoming H.323 call, Cisco Unified Border Element (SP Edition) carries out routing to determine the next hop for the call.

SBC policy allows calls to be routed to one of the following:

- signaling peer (such as a gateway)
- outgoing gatekeeper

  When a gatekeeper is used, the gatekeeper is responsible for resolving the called party number to a next hop address.

In a SBC configuration, a routing next hop is identified by an adjacency name. The adjacency is configured with the address of the next hop gateway or gatekeeper.

Support for H.323 addressing

All H.323 calls through Cisco Unified Border Element (SP Edition) need to specify a called party number. A called party number may optionally be supplied in the Q.931 calledPartyNumber or the H.225 destinationAddress, with the former taking priority. If a called party number is not present in either of these fields, then the SBC rejects the call.

Finally, the connected number may also optionally be supplied in the Q.931 connectedNumber or the H.225 connectedAddress, with the former taking priority. The connected number indicates the party the call ends up connecting with because during call setup, the call might be redirected or the called number might be edited along the way.

When an H.323 endpoint sends out a Q.931/H.225 message, the called and calling numbers are always placed in the Q.931 fields, not the H.225 fields.

DNS Name Resolution

Domain name server (DNS) name resolution enables you to use the domain name instead of the IP address in an adjacency configuration. You can configure both gatekeeper and non-gatekeeper adjacencies with DNS names.

If you use a DNS name in an adjacency configuration, the name is resolved each time a call is routed out over that adjacency. This process allows DNS-based load-balancing.

Number Validation and Editing

Cisco Unified Border Element (SP Edition) allows validation, editing and categorization of the called and calling party number through a Number Validation configuration.

This can be used for comparing or editing of source or destination telephone numbers or textual usernames. This process is called Number Analysis (NA). Number Analysis (NA) determines whether a set of source or destination digits, or source or destination textual addresses represents a valid address (based on number validation, number categorization, and/or digit manipulation). This is achieved by configuring one or more tables of valid addresses and editing rules in the tables. Matching for digit strings uses a limited-form of specialized regular-expression syntax and matching for textual addresses is done on the basis of the Basic Regular Expression syntax. In both cases, either the entire address or part of the address can be matched.

NA can be optionally configured as a step within the call policy set.

For more information, see the ?$paranum>Number Analysis Policies? section on page 7-14 section and the ?$paranum>Number Analysis? section on page 7-6 in the Implementing Cisco Unified Border Element (SP Edition) Policies chapter.

Load Balancing

Cisco Unified Border Element (SP Edition) can load balance between H.323 adjacencies using Round Robin or Least Cost Routing configurations.

Round Robin load balancing distributes calls evenly between adjacencies. Least Cost load balancing assigns a priority to each adjacency.

For example, routing might route two consecutive calls onto two different adjacencies.

- For gatekeeper adjacencies, the calls will be admitted on two different gatekeepers. It is up to the gatekeeper routing configuration to determine whether the signaling next hop for each call is the same.
- For non-gatekeeper adjacencies, the signaling next hop will be set to two different gateways (or terminals).

If a gatekeeper adjacency loses contact with the gatekeeper, it is temporarily taken out of service - meaning that the SBC will not attempt to route new calls through it. If there is an alternative route, call setup will continue on the alternative route. You can also manually deactivate an adjacency, which has the same effect.

Inter-VPN Calling

Cisco Unified Border Element (SP Edition) can peer with H.323 devices in different VPNs simultaneously.

You configure VPNs on a per-adjacency basis. Therefore, inter-VPN calling is simply a matter of your configuring a routing policy that routes calls between adjacencies in different VPNs.

Call Admission Control

This section describes the following:

- Privacy Service, page 7-47

Call Admission Control Overview

Call Admission Control (CAC) allows you to configure policy for accepting or rejecting calls. It allows you to apply detailed policies to certain call options to limit the number of concurrent calls and registrations. CAC can restrict the media bandwidth dedicated to active calls. It allows for load control on other network elements by rate limiting. Certain events can be completely blocked (using a blacklist) or freely allowed (using a whitelist), based on certain attributes.

CAC determines whether an event should be granted or refused based on configured limits for network resource utilization. There are two reasons for performing call admission control.

- To defend load-sensitive network elements, such as softswitches, against potentially harmful levels of load precipitated by singular events, such as DoS attacks, natural or man-made disasters, or mass-media phone-ins.
- To police the Service Level Agreements (SLAs) between organizations, to ensure that the levels of network utilization defined in the SLA are not exceeded.

Call admission control is the final stage of the call policy, so it is applied after number analysis and routing policy. CAC policy is applied to all event types, such as new calls, subscriber registrations, and call updates. If an event is not granted by the CAC policy, then Cisco Unified Border Element (SP Edition) rejects it with a suitable error code.

A CAC policy consists of the following.

- A limit or limits that must not be exceeded.

  Limits, for example, can be set on the maximum number of concurrent calls, the maximum rate of calls, or the maximum bandwidth consumed by calls.

- A scope at which the limits are applied.

  This can be global, per-account, per-adjacency, or any of the scopes defined in Policy scopes. Combinations of scopes can also be used, such as “per account, per number category.” Scope is part of the policy itself. For example, in the policy “maximum 20Kb per call,” the scope is “per call.”

To define an admission control policy, you must define the limit and the scope at which it is applied. For example, you can define a policy such that not more than 10 concurrent calls (limit) could ever be made from a single account (scope).

Although the scope and limits define the policy, they do not determine when the policy is applied. For example, you cannot name a particular account, such as “account1,” as the scope for your policy. Instead, the table-type and match value are used to determine when a policy is applied. Setting “account” as the table-type and “account1” as the match value matches call events from account1.

Compound Scopes

Compound scopes provide a more elaborate set of options for configuring policy. Certain policy scopes can be combined to create compound scopes. To combine scopes, configure each scope using a separate **first-cac-scope** or **cac-scope** command.

The following are examples of compound scopes:

- If you want to restrict the number of calls between any pair of adjacencies to 20, you could create a policy with MaxCalls = 20 and a scope of “src_adjacency, dst_adjacency.” This policy would restrict the number of calls between any pair of adjacencies to 20. However, it would not limit the total number of calls out of any adjacency, nor the total number of calls into any adjacency.

- You can define an admission control policy at a compound scope of "source adjacency and category," and set the maximum concurrent calls in this scope to 10. This policy would restrict the number of concurrent calls of the same category that each adjacency could make to 10. The scope field value is src-adjacency, category.

Policy Scopes

Table 7-4 defines the scopes in which call admission policies can be applied and specifies whether each of these scopes can be combined with other scopes.

Table 7-4 Policy Scope Definitions

Scope Option or Value of Scope Field	Scope	Description	Can Scope Be Combined?
account	Per account	The limits specified in this scope apply to all the events from the same account.	Yes, except the dst-account and src-account scopes
adjacency	Per adjacency	The limits specified in this scope apply to all the events from the same adjacency.	Yes, except the src-adjacency, dst-adjacency, src-adj-group, and dst-adj-group scopes
adj-group	Per adjacency group	The limits specified in this scope apply to all events sent to or received from the same adjacency group. For example, you can restrict the total number of concurrent calls that can be sent to or received from the adjacencies in a single adjacency group by configuring limits in this scope.	Yes, except the adjacency, src-adj-group, and dst-adj-group scopes
call	Per call	The limits specified in this scope apply to any single call. For example, you can restrict the per-call bandwidth or the allowed call update rate by configuring limits in this scope. Note that some limits are invalid in this scope.	No
category	Per category	The limits specified in this scope apply to all events that have been placed in the same category by the number analysis policy tables. For example, you can restrict the total number of concurrent calls in any single category by configuring limits in this scope.	Yes
dst-account	Per destination account	The limits specified in this scope apply to all events sent to the same account. For example, you can restrict the total number of concurrent calls that can be sent to any single account by configuring limits in this scope.	Yes, except the account scope

Table 7-4 Policy Scope Definitions (continued)

Scope Option or Value of Scope Field	Scope	Description	Can Scope Be Combined?
dst-adj-group	Per destination adjacency group	The limits specified in this scope apply to all events sent to the same adjacency group. For example, you can restrict the total number of concurrent calls that can be sent to the adjacencies in a single adjacency group by configuring limits in this scope.	Yes, except the adj-group scope
dst-adjacency	Per destination adjacency	The limits specified in this scope apply to all events sent to the same adjacency. For example, you can restrict the total number of concurrent calls that can be sent to any single adjacency by configuring limits in this scope.	Yes, except the adjacency scope
dst-number	Per dialed number	The limits specified in this scope apply to all events that have the same destination number. For example, you can restrict the total number of concurrent calls to any single valid number by configuring limits in this scope.	Yes
global	Global	The limits specified in this scope apply to SBC as a whole.	No
src-account	Per source account	The limits specified in this scope apply to all events received from the same account. For example, you can restrict the total number of concurrent calls that can be initiated from any single account by configuring limits in this scope.	Yes, except the account scope
src-adj-group	Per source adjacency group	The limits specified in this scope apply to all events received from the same adjacency group. For example, you can restrict the total number of concurrent calls that can be initiated from the adjacencies in a single adjacency group by configuring limits in this scope.	Yes, except the adjacency and adj-group scopes
src-adjacency	Per source adjacency	The limits specified in this scope apply to all events received from the same adjacency. For example, you can restrict the total number of concurrent calls that can be initiated from any single adjacency by configuring limits in this scope.	Yes, except the adjacency scope
src-number	Per dialing number	The limits specified in this scope apply to all events that have the same source number. For example, you can restrict the total number of concurrent calls from every single source number by configuring limits in this scope.	Yes

Table 7-4 Policy Scope Definitions (continued)

Scope Option or Value of Scope Field	Scope	Description	Can Scope Be Combined?
sub-category	Per subscriber category	**Note** This is not supported in Cisco IOS XE Release 2.4. The limits specified in this scope apply to all events sent to or received from members of the same subscriber category. For example, you can restrict the total number of concurrent calls that can be sent to or received from the subscribers in a single subscriber category by configuring limits in this scope.	Yes, except the sub-category-pfx and subscriber scopes
sub-category-pfx	Per subscriber category prefix	**Note** This is not supported in Cisco IOS XE Release 2.4. The limits specified in this scope apply to all events sent to or received from members of the same subscriber category prefix. For example, you can restrict the total number of concurrent calls that can be sent to or received from the subscribers in a single subscriber category prefix by configuring limits in this scope.	Yes, except the sub-category-pfx and subscriber scopes
subscriber	Per subscriber	**Note** This is not supported in Cisco IOS XE Release 2.4. The limits specified in this scope apply to all events sent to or received from individual subscribers. A subscriber is any device in the network that has registered with a Registrar server via SBC, or with an S-CSCF in an IP Multimedia Subsystem (IMS) network. This does not allow you to match on a specific subscriber.	Yes, except the sub-category-pfx and subscriber scopes

Note If you are supporting Aggregate Registrations in a non-IMS network, all of the phones behind a device (such as a PBX) are counted as the same subscriber if you are using a per-subscriber scope.

Non-Subscriber Group

When a subscriber scope is enabled, the SBC includes an additional group of ALL “non-subscribers.” The non-subscribers are counted within a special group of the subscriber scope. The non-subscriber group is matched if the call is from a non-subscriber. Limits set in the subscriber scope apply to this non-subscriber group.

Note A "subscriber" is identified using the Address-of-Record that is registered with the registrar. A "subscriber category" is based on the source IP address of the SIP message. When some subscribers sit behind a Network Address Translation (NAT) device and share the same IP address, they are in the same subscriber category. However, they differ among each other by their AOR.

Policy Set Tables and Limit Tables

Call admission control policies are configured using a combination of Policy Set and Limit tables.

A Policy Set table type is applied to all entries defined within the CAC table. Each entry within the table configures its own scope. Every entry in a Policy Set table automatically matches every event that reaches that table. Policy Set tables create multiple policies for each event.

A Limit table type selects the single best matching match value defined in a CAC entry. The scope for the limit table type is inherited from the limit table's parent table. The entries in a Limit table specify the values to match against and the limits to apply if a match is achieved.

The major difference between a Policy Set table and a Limit table is that the Policy Set table creates multiple policies for a given event, while a Limit table only defines one policy for a given event.

For information on table-types, match values, and when an event matches an entry for Limit Table, see Table 7-5. For information on scope name, scope definition, and whether a scope can be combined, see Table 7-4.

Limit Tables

Table 7-5 lists the types of Limit tables. For each table type, the corresponding Match value is listed, with the conditions under which a match is achieved. If a match is achieved, the corresponding policy is applied to the event.

Table 7-5 Table Types for Limit Table

Table Type	Match Value	Conditions Where an Event Matches an Entry
account	account name	Match value is the source and/or destination account name.
adj-group	adjacency group name	Match value is the source and/or destination adjacency group name.
adjacency	adjacency name	Match value is the source and/or destination adjacency name.
all	NA	All events match entry
call-priority	SBC priority	SBC priority is the event call-priority.
category	category name (assigned during number analysis)	Event has been assigned a category, and match value is the name of the category assigned.
dst-account	account name	Match value is the destination account name.
dst-adj-group	adjacency group name	Match value is the destination adjacency group name.
dst-adjacency	adjacency name	Match value is the destination adjacency name.

Table 7-5 Table Types for Limit Table (continued)

Table Type	Match Value	Conditions Where an Event Matches an Entry
dst-prefix	number prefix	Match value is the first digits of the number being called.
event-type	Type of event to which CAC policy is applied (new-call, call-update or endpoint-reg)	Match value is the event type.
src-account	account name	Match value is the source account name.
src-adj-group	adjacency group name	Match value is the source adjacency group name.
src-adjacency	adjacency name	Match value is the source adjacency name.
src-prefix	number prefix	Match value is the first digits of the calling number
sub-category	ipv4 {*ip-address*} [vrf *vrf*]	Match value is the IPv4 address. When the "sub-category" table type is defined for a CAC table, you must define the match-value within the entry. As an example, you would use the command: **match-value ipv4** {*ip-address*} [vrf *vrf*]
sub-category-pfx *pfx-len*	ipv4 {*ip-address*} {*prefix-len*} [vrf *vrf*]	Match value is the IPv4 address. When the "sub-category-pfx *pfx-len*" table type is defined for a CAC table, you must define the match-value and match-prefix-len within the entry. As an example, you would use the command: **match-value ipv4** {*ip-address*} {*prefix-len*} [vrf *vrf*].

CAC Table Entry Configuration Commands

Each CAC table consists of a collection of table entries, defined within the CAC table submode. For Policy Set table types, the CAC scope is defined within each entry. If unspecified, the scope defaults to global for that entry.

For Limit table types, the CAC entry specifies a value to match against. The semantics of this match-value are determined by the type of Limit table.

For both table types, the limits defined within the entry are calculated using per scope values. Some limits are not applicable at all scopes. Policy Set table types define the scope within the entry, thus both the limit and the scope are per entry. If you want per entry limits for a Limit table type, then configure the Limit table type to match the scope.

See the ?$paranum>Configuring Call Admission Control Policy Sets, CAC Tables, and Global CAC Policy Sets? section on page 7-115 for detailed configuration step information.

Table 7-6 shows a list of various limits and options that can be configured on an entry in a CAC policy-set table. These configurable command options can be displayed with the following commands:

```
Router(config-sbc-sbe-cacpolicy-cactable-entry)# cac-table 4
Router(config-sbc-sbe-cacpolicy-cactable)# table-type policy-set
Router(config-sbc-sbe-cacpolicy-cactable)# entry 1
Router(config-sbc-sbe-cacpolicy-cactable-entry)# ?
```

Note The cac-scope command option is only displayed for Policy Set table types. The match-value command option is only displayed for Limit table types.

Table 7-6 CAC Table Entry Configurable Command Options

Configurable Command Option	Description
cac-scope	Scope at which CAC limits are applied within each entry in a Policy Set table.
callee	Callee settings
callee-codec-list	List of codecs which the callee leg of a call is allowed to use
callee-hold-setting	The callee hold setting supported
callee-inbound-policy	Set callee inbound Session Description Protocol (SDP) policy table
callee-outbound-policy	Set callee outbound SDP policy table
callee-privacy	The level of privacy processing
callee-sig-qos-profile	QoS profile to use for callee signalling
callee-video-qos-profile	QoS profile to use for callee video media
callee-voice-qos-profile	QoS profile to use for callee voice media
caller	Caller settings
caller-codec-list	List of codecs which the caller leg of a call is allowed to use
caller-hold-setting	The caller hold setting supported
caller-inbound-policy	Set caller inbound sdp policy table
caller-outbound-policy	Set caller outbound sdp policy table
caller-privacy	the level of privacy processing
caller-sig-qos-profile	QoS profile to use for caller signalling
caller-video-qos-profile	QoS profile to use for caller video media
caller-voice-qos-profile	QoS profile to use for caller voice media
codec-restrict-to-list	Restrict to using codecs from a configured codec list
early-media-deny	Do not allow early-media
early-media-timeout	Duration for which to allow early media
early-media-type	Directions in which to allow early media
match-value	Match-value of an entry in a CAC Limit table
max-bandwidth	Maximum bandwidth
max-call-rate-per-scope	Maximum call rate
max-channels	Maximum number of channels
max-in-call-msg-rate	Configure maximum rate of in-call messages. See description of in-call messages in the ?$paranum>CAC Rate Limiting? section on page 7-45.
max-num-calls	Maximum number of calls

Table 7-6 CAC Table Entry Configurable Command Options (continued)

Configurable Command Option	Description
max-out-call-msg-rate	Configure maximum rate of out-of-call messages
max-regs	Maximum subscriber registrations
max-regs-rate-per-scope	Maximum subscriber registrations rate
max-updates	Maximum updates to call media
media	Media Flag
transcode-deny	Sets transcoding to forbidden for the admission control entry
transport	Transport Protocol Parameters

Nonlimiting CAC Options

CAC allows you to configure policy for accepting or rejecting calls based on limit options such as max-num-calls and max-bandwidth. The CAC scope is used when policing limit options. CAC also allows you to apply a property to a call (rather than a limitation) with nonlimiting options, such as caller-inbound-policy. Scopes have no meaning for nonlimiting options.

You can configure multiple CAC policies that all apply to a given event (using a Policy Set table type). A nonlimiting option can be given contradictory values in each of these policies. CAC determines what its behavior towards that event is by examining the setting of the option in each applicable policy and applying a rule to produce a "derived value" for the field. If the option is not defined in any policy, then a default behavior is defined. When the SBC is deriving a value for a nonlimiting field, it should disregard all policies in which that field has not been defined by the user. The SBC derives that value based on the assigned behavior for the specific nonlimiting option. The behavior for the nonlimiting options takes one of the following values:

- Last non-default value used. Options of this type take the last non-default value as the derived value. For example, caller-inbound-policy uses the last found non-zero length sdp policy name as the derived value.
- Most restrictive value used. Options of this type take as the derived value the Policy Value that most restricts the behavior of the SBC.
- First non-default value used. Options of this type use the first non-default value as the derived value. For example, caller-voice-qos-profile uses the first non-zero length voice QoS profile name as the derived value.
- All found values combined. Options of this type perform a bitwise-OR to obtain a cumulative value as the derived value.

Table 7-7 Nonlimiting Options in CAC Entries

Nonlimiting Option in a CAC Entry	Behavior of Derived Value
branch bandwidth-field	Last nondefault value used
branch codec-list	Last nondefault value used
branch hold-setting	Last nondefault value used
branch inbound-policy	Last nondefault value used
branch media-description	All found values combined
branch media-type	Last nondefault value used

Table 7-7 Nonlimiting Options in CAC Entries (continued)

Nonlimiting Option in a CAC Entry	Behavior of Derived Value
branch outbound-policy	Last nondefault value used
branch privacy	Most restrictive value used
branch secure-media	All found values combined
branch sig-qos-profile	First nondefault value used
branch tel-event payload type	Last nondefault value used
branch video-qos-profile	First nondefault value used
branch voice-qos-profile	First nondefault value used
callee-bandwidth-field	Last nondefault value used
callee-codec-list	Last nondefault value used
callee-hold-setting	Last nondefault value used
callee-inbound-policy	Last nondefault value used
callee media-description, callee secure media	All found values combined
callee media-type	Last nondefault value used
callee-outbound-policy	Last nondefault value used
callee-privacy	Most restrictive value used
callee-sig-qos-profile	First nondefault value used
callee tel-event payload type	Last nondefault value used
callee-video-qos-profile	First nondefault value used
callee-voice-qos-profile	First nondefault value used
caller-bandwidth-field	Last nondefault value used
caller-codec-list	Last nondefault value used
caller-hold-setting	Last nondefault value used
caller-inbound-policy	Last nondefault value used
caller media-description, caller secure media	All found values combined
caller media-type	Last nondefault value used
caller-outbound-policy	Last nondefault value used
caller-privacy	Most restrictive value used
caller-sig-qos-profile	First nondefault value used
caller tel-event payload type	Last nondefault value used
caller-video-qos-profile	First nondefault value used
caller-voice-qos-profile	First nondefault value used
codec-restrict-to-list	Last nondefault value used
early-media-deny	Most restrictive value used
early-media-timeout	Most restrictive value used
early-media-type	Most restrictive value used

Table 7-7 Nonlimiting Options in CAC Entries (continued)

Nonlimiting Option in a CAC Entry	Behavior of Derived Value
media address preserve, media bandwidth-field ignore, media tel-event interworking	All found values combined
sdp-media-profile	Last nondefault value used
transcode-deny	Most restrictive value used
transport srtp	Most restrictive value used

Configuring Directed Nonlimiting CAC Policies

In releases prior to Release 3.5.0, you can use the **caller** command and the **callee** command to configure the CAC policy entries that are applied when an adjacency, adjacency group, or account is either a caller or a callee in a call. However, this approach does not permit the configuration of certain directed nonlimiting CAC policy fields on specific adjacencies, adjacency groups, or accounts, in a way that is independent of whether the adjacencies, adjacency groups or accounts are the callees or the callers on the calls. The following example illustrates this limitation.

Suppose the following sequence of commands is part of the configuration of an entry in a CAC table:

```
cac-policy-set 3
    first-cac-table cac-tb1
    cac-table cac-tb1
     table-type limit adjacency
     entry 1
      match-value adj1
      caller port-range-tag adj-name
      callee port-range-tag adj-name
      action cac-complete
```

If there is a call from the adj1 adjacency to the adj2 adjacency, the settings specified for the caller in this example is applied to adj1. At the same time, the callee settings are applied to adj2 because that adjacency is the callee in this call. In a scenario such as this one, you might not want to apply any configuration to the other adjacency (the adj2 adjacency, in this example) involved in the call. The **branch** command helps overcome this limitation. This command has been introduced in Release 3.5.0.

In the preceding example, the **branch** command can be used to replace the **caller** command and the **callee** command as follows:

```
cac-policy-set 3
    first-cac-table cac-tb1
    cac-table cac-tb1
     table-type limit adjacency
     entry 1
      match-value adj1
      branch port-range-tag adj-name
      action cac-complete
```

Note The **branch** command is not a replacement for the **caller** command and the **callee** command pair in scenarios in which you want to apply settings to both the caller adjacency and the callee adjacency.

With this configuration, the settings specified in the **branch** command are applied to the adj1 adjacency. For a call from the adj2 adjacency to the adj1 adjacency, the same settings are applied to the adj1 adjacency. For this call, no settings are applied to adj2 or any other adjacency that calls or is called by adj1.

The following are the features of the **branch** command:

- If a branch setting (that is, the **branch** command) and a caller-callee pair setting (that is, the **caller** and **callee** command pair) are configured in different policy entries, the setting in the last entry of the configuration takes precedence.
- If two branch settings, each in a different policy entry, are encountered, the setting in the last entry that is encountered takes precedence.
- If a branch setting and a caller-callee setting are in the same policy entry, the branch setting takes precedence over the caller-callee setting.

  The following sample configuration illustrates how the branch command works:

```
cac-policy-set 3
    first-cac-table cac-tb1
    cac-table cac-tb1
     table-type limit adjacency
     entry 1
      match-value phone2
      branch port-range-tag adj-name
      caller port-range-tag string tagB_cac
      callee port-range-tag string tagA_cac
      action cac-complete
     entry 2
      match-value phone1
      branch port-range-tag string tagA_cac
      caller port-range-tag adj-name
      callee port-range-tag adj-name
      action cac-complete
    complete
   cac-policy-set global 3

media-address ipv4 209.165.202.130
  port-range 10000 15000 any
  port-range 15002 15003 any tag phone1
  port-range 16002 16003 any tag phone2
  port-range 17002 17003 any tag tagA_cac
  port-range 18002 18003 any tag tagB_cac
```

  In this example, the call goes from phone 1 to phone 2. The following sequence of events takes place during the call:

1. Matching is performed on the source adjacency, phone 1, which matches entry 2. Here, the branch entry refers to the caller side, so the caller entry is overridden. After this first policy match is performed, port-range-tag is set to tagA_cac on side A. In addition, the callee port tag is set to adj-name.
2. Matching is performed on the destination adjacency, phone2, which matches entry 1. Here, the branch entry refers to the callee side, so the callee entry is overridden. This entry sets the caller side port-range-tag to tagB_cac. In other words, adj_name is assigned as the callee side port-range-tag. These settings take precedence over the values assigned in the previously matched entry, entry 2, because these settings are assigned later.

The outcome is that the tagB_CAC port is used on side A, and an adj-name port, phone2, is used on side B.

Media Line Removal

Media line removal feature provides the ability to strip or pad disabled media descriptions (m-lines with zero port) when sending an offer or answer to interoperate with various non-compliant devices.

Where the SDP being forwarded represents an answer, the media line which was removed from the forwarded offer is identified and a dummy media line is inserted into the same location. This is required for the compliant partner to match appropriate media line requests and responses.

Where the SDP being forwarded is a future offer, it uses offer modification to effectively shuffle-up media lines allowing the "padding" dummy media lines to be added to the end of the forwarded SDP.

SBC's transmit behavior is independently configured for the caller and callee sides of the call using the following options:

- strip new on offer—removes disabled media streams in forwarded offers which are new or unknown to the recipient of the offer.
- strip all on offer—removes all disabled media streams from forwarded offers, whether known to the recipient of the offer or not.
- strip on answer—removes all disabled media streams from forwarded answers.
- do not pad on offer—stops SBC from padding forwarded offers with disabled media streams. This means that a forwarded offer may not comply because it may contain less media lines than previous offers.

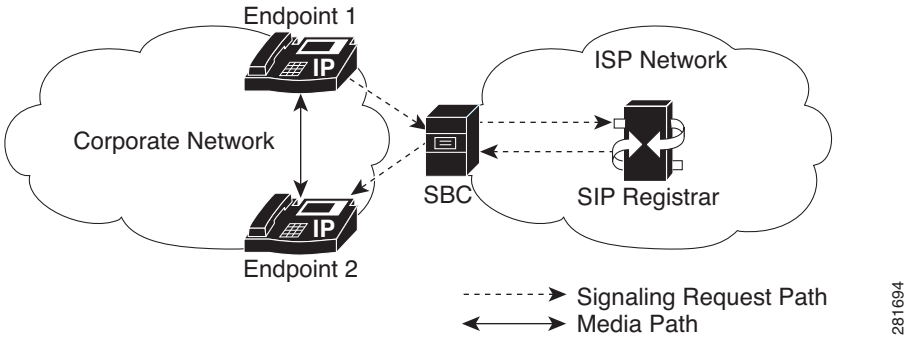
Note The "strip new on offer" and "strip all on offer" result in removal of m-lines from the forwarded offer. The missing lines are not "padded in" and there is no need to set the "do not pad on offer" option to achieve this. The "do not pad on offer" option only affects media lines that were missing from the received offer.

On selecting the appropriate option, the SDP to be forwarded is created with disabled media portions deleted, rather than the existing behavior of setting the port to zero.

Multiple SBC Media Bypass

The multiple SBC media bypass feature can send media packets directly from the answerer to the original offerer. When the SBC detects that the media packets are being looped back unnecessarily, as shown in Figure 7-5, the SBC removes itself from the loop so that the media packets can flow directly between the endpoints.

Figure 7-5 Loopback Signaling Across The Same SBC—Two Call Media Bypass



Partial Media Bypass

When at least one SBC from the network has to anchor the media because endpoints cannot communicate directly, the other SBC gets bypassed as shown in Figure 7-6. If the media bypass type is explicitly configured to be partial, only IP realm and VPN configuration on the adjacency can be used to determine whether media bypass is possible. Because media bypass tags are not used, the VPN names must be globally unique across all the SBCs for partial media bypass to work.

Figure 7-6 Partial Media Bypass

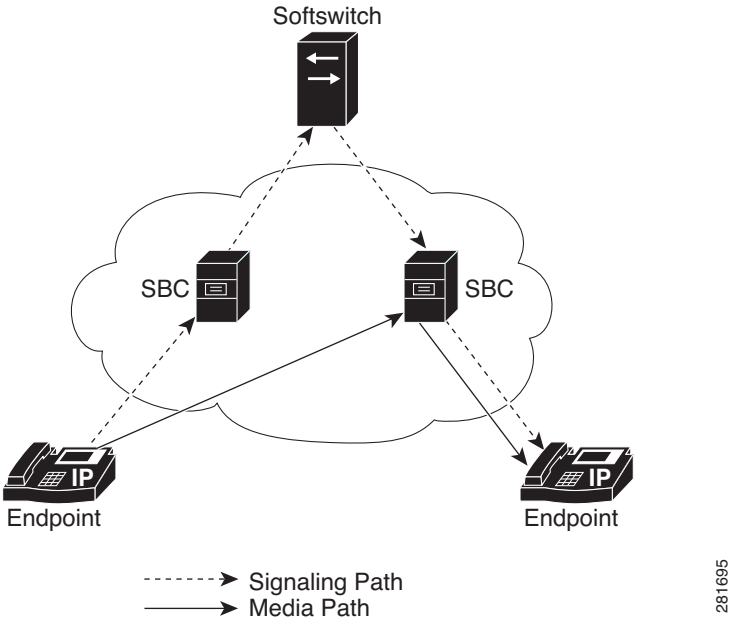
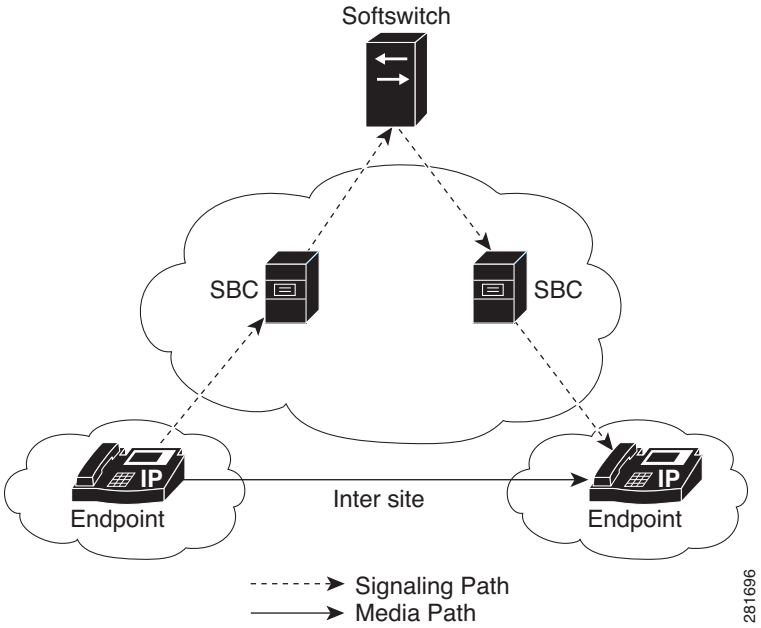


Figure 7-7 shows an example of media bypass across two or more SBC devices

Figure 7-7 Multiple SBC Media Bypass



In networks where direct media packets cannot pass, the feature creates an optimized media path through a group of SBCs, to avoid unnecessary media hops through the SBC network.

With the multiple SBC media bypass feature, the SBC can transmit an extra set of media addresses alongside an SDP offer. These are the original media addresses that the SBC itself received from the offerer. The original media addresses are placed in a separate multiple SBC media bypass feature information element. These addresses are associated with information about the media plane connectivity of the offerer. A downstream SBC uses the multiple SBC media bypass feature connectivity information to determine whether it can re-instate the original media addresses by rewriting the SDP offer to include them. This enables the media packets to directly pass between the answerer and the original offerer.

The multiple SBC media bypass feature information can also be used by a group of SBCs to optimize the media path and to avoid unnecessary media hops through the SBC network. The SDP answer is accompanied by an indication of whether the feature was successful or not. The SBC uses this indication to determine whether it has been bypassed or whether it is still in the media path. When many SBCs appear in the media path, they collectively build up a stack of alternative media addresses for each media streams in the offer, where each element of the stack has associated connectivity information.

The SBCs determines which endpoints and intermediate hops are connected to decide which intermediate entities can be bypassed. Such connectivity information is passed on by tags in the multiple SBC media bypass feature information elements. Two remote endpoints on an adjacency can be connected if they have one or more matching tags. Therefore, tags must be globally unique for the multiple SBC media bypass feature protocol to work.

Continuing Media Bypass After a Session Refresh

When a media bypass call is in progress, the SIP registrar does not process the media exchanged by the endpoints. Therefore, the registrar uses signaling to detect failures in the session. The registrar sends a session refresh request to check whether a session is alive. The session refresh request is in the form of an INVITE or UPDATE message containing a copy of the SDP forwarded by the registrar during the original call setup.

When the SBC receives the INVITE message from the SIP registrar, it does not correlate the SDP in the message with the SDP sent earlier in the call. The SBC processes the SDP in the INVITE message as normal and creates an SDP offer to send to either the caller endpoint or callee endpoint. From the perspective of the endpoint, the INVITE message is an attempt to renegotiate the media for the call. The endpoint processes the offer and creates an answer that is consistent with the offer. This answer is returned to the registrar through the SBC.

In the answer, the port number for each media stream in the call is different from the port number of the previous media stream. This mismatch in the port number could cause the registrar to send a late INVITE message to the endpoint. An endpoint that does not support the receipt of a late INVITE message for a renegotiation would reject the message. The call fails because the media is being sent from the endpoint to the SBC, from where the media is dropped. To circumvent this issue, renegotiation is enabled by default so that the same path is used to resume exchange of media packets between the endpoints. This feature ensures that media bypass calls continue to bypass the media after a session refresh.

Note You can disable or enable renegotiation.

Restrictions

The multiple SBC media bypass feature has the following restrictions:

- Media bypass is not supported for H.323 calls.
- Media services, such as provisioned transcoding, transrating, and DTMF Interworking preclude media, are sent directly between endpoints. For a given call, if the administrator has configured media bypass settings and if media bypass is possible, then it takes precedence over other media services. However, if lawful intercept (LI) is provisioned on the SBC, LI would take precedence over the multiple SBC media bypass feature.
- The SBC does not support the feature when one endpoint is IPv4 and the other endpoint is IPv6. Because the endpoints cannot understand the traffic they receive.
- The SBC does not support the feature when one endpoint is SIP and the other endpoint is H.323 if SIP-H.323 interworking is enabled. Because the endpoints cannot understand the traffic they receive.

Performance Impact

When the multiple SBC media bypass feature is enabled, it has the following performance impact on the Cisco ASR 1000 series routers:

- The SBCs signaling performance decreases by a small fraction, due to the increased parsing and message manipulation costs. However there is a corresponding gain on media resources for every call that successfully negotiates the feature.
- The transient occupancy of each call setup increases by a multiple of the size of the multiple SBC media bypass feature information that is encoded in the SIP message, plus a small amount of control information. For SDP sizes of 300 bytes, this is predicted to be around 1500 bytes in total. However,

the steady-state occupancy for calls that successfully negotiate the multiple SBC media bypass feature decreases as no media resources are required for those calls. This saves approximately 10000 bytes per call.

For more information on configuring the multiple SBC media bypass feature, see the ?$paranum>Configuring Multiple SBC Media Bypass? section on page 7-136. For configuration examples of the feature, see the ?$paranum>Example: Multiple SBC Media Bypass? section on page 7-161. For video of the example that explains how the SBC Media Bypass feature works, see http://www.cisco.com/en/US/docs/routers/asr1000/configuration/guide/SBCU3.5S/sbc_media_bypass.html.

Common IP Address Media Bypass

This section contains the following topics:

Restrictions for Common IP Address Media Bypass

The following are restrictions for the Common IP Address Media Bypass feature:

- This feature is not supported for H.323 adjacencies.
- This feature is not supported in a scenario in which endpoints are behind the same NAT device but are not registered with the SBC.
- This feature is not supported in a scenario in which the caller endpoint and callee endpoint are behind different NAT devices even when there is connectivity between the networks defined by each NAT device. The SBC always relays media between two such endpoints.
- This feature is not supported in a scenario in which only one of the endpoints is behind a NAT device. The SBC always relays media between two such endpoints.

Information About Common IP Address Media Bypass

When you enable the Multiple SBC Media Bypass feature, the SBC bypasses or relays media between the endpoints of an adjacency depending on the media bypass tags presented by the endpoints. If the tags match, the SBC determines that there is media connectivity between the endpoints and, therefore, bypasses itself from the media flow between the endpoints. In contrast, if the tags do not match, the SBC determines that there is no media connectivity between the endpoints and, therefore, relays media between the endpoints.

Note For detailed information about the Multiple SBC Media Bypass feature, see the ?$paranum>Multiple SBC Media Bypass? section on page 7-39.

An organization can use a hosted PBX solution that is owned and managed by a service provider. Typically, a hosted PBX solution can serve many organizations and, therefore, serve multiple NAT devices. There may be a scenario in which there are multiple NAT devices behind a single adjacency. In such a scenario, the SBC must bypass media for the endpoints behind the same NAT device and relay media for the endpoints that are behind different NAT devices. In releases prior to Release 3.6.0, the only way to achieve this is to configure an adjacency for each NAT device. This approach increases the overhead involved in managing the network.

The Common IP Address Media Bypass feature is an enhancement to the Multiple SBC Media Bypass feature. It offers an alternative to the approach of creating an adjacency for each NAT device.

When you configure the Common IP Address Media Bypass feature, the SBC assigns each endpoint behind a NAT device a media bypass tag that is based on the corresponding endpoint's external, NAT IP address. These media bypass tags are used by the SBC to determine whether the caller endpoint and callee endpoint belong to the same NAT network. If the media bypass tag of the caller endpoint matches the media bypass tag of the callee endpoint, the SBC bypasses media. If the tags do not match, the SBC relays media.

Note The Common IP Address Media Bypass feature does not introduce any change in the mechanism by which the SBC compares the media bypass tags of the caller endpoint and callee endpoint. In other words, the SBC does not distinguish between the feature or method by which media bypass tags are created. The SBC only compares the tags and bypasses media when the tags match.

When the Common IP Address Media Bypass feature is not configured or is disabled, media-bypass decisions are taken by the SBC on the basis of the media bypass tags configured at the adjacency level. If media bypass tags are not configured, media-bypass decisions are taken on the basis of the autogenerated tags that are based on VPN IDs.

When the Common IP Address Media Bypass feature is configured and enabled:

- If the caller endpoint or callee endpoint is a registered subscriber that has been identified at registration time as being behind a NAT device, the SBC generates a media bypass tag and uses that media bypass tag for the call leg.
- If the caller endpoint or callee endpoint is not a registered subscriber or is not behind a NAT device, the SBC uses the tag that is created by the **media bypass tag** command, if such a tag is present, for the call leg.
- If the caller endpoint or callee endpoint is not a registered subscriber or is not behind a NAT device and if there are no configured tags, the SBC generates a media bypass tag based on the VPN ID of the endpoint, for the call leg.

During the call, if both the endpoints have media bypass tags that match, the SBC determines that both the endpoints are behind the same NAT device and it bypasses media for that call leg. Conversely, if the media bypass tags do not match, if either endpoint does not have a media bypass tag, or if either endpoint is not a registered subscriber, the SBC relays media for that call leg.

The following is the format of the media bypass tag generated by this feature:

nat-*VPN-ID-IP-address*

In this format, *IP-address* is the source IP address of the most recent non-fast-pathed, successful REGISTER request from the endpoint. The IP address can be in IPv4 format or IPv6 format.

The following are sample media bypass tags generated by this feature:

- nat-123-192.0.2.6
- nat-254-192.0.26.18
- nat-2233-2001:DB8::AC10:FE01

Features of Common IP Address Media Bypass

The following are additional points about how the Common IP Address Media Bypass feature works:

- After this feature is configured, the SBC can detect whether an endpoint is behind a NAT device by using the existing adjacency configuration features:

  - If the **no nat** command is configured for an adjacency, an endpoint behind that adjacency is recognized as being behind a NAT if the IP address in the Via header of the SIP messages from that adjacency is different from the IP address from which the request was received.
  - If the **nat force-on** command is configured, all endpoints are assumed to be behind a NAT.
  - If the **nat force-on** command is configured and an endpoint is not behind a NAT, the SBC relays media for calls to and from such an endpoint. Note that if this feature is disabled, the SBC bypasses media. When you enable this feature, the SBC starts relaying media.
- If this feature is configured while an adjacency is active, only new calls that are processed by that adjacency are affected by this feature. Existing calls are not affected.
- This feature is independent of whether the initial INVITE message contains SDP content because the media bypass tag is not added to the SDP content.
- This feature is supported by all forms of media bypass:
  - Simple media bypass, in which the caller endpoint and callee endpoint are associated with local adjacencies.
  - Two-call media bypass, in which the SBC forwards the call to a softswitch or registrar, which then loops the call request back to the SBC.
  - N-call media bypass, in which a call is looped through the SBC multiple times.
  - Multi-SBC media bypass, in which a call is looped through multiple SBCs.

CAC Rate Limiting

You can limit the number or the rate of new calls accepted and the number of media renegotiations within a call. However, limits are not placed on the following:

- Media renegotiations which do not actually change the characteristics of the call.
- Any other in-call messages.

  In-call messages include any message within the context of a call, including provisional responses during call setup and call renegotiation messages, but not including call setup or tear-down messages.
- Internally-generated messages

Note You cannot specify limits at the granularity of a specific SIP or H.323 message.

You can also limit the rate and number of registrations passing through the Cisco Unified Border Element (SP Edition). However, limits are not placed on any other out-of-call messages. (An out-of-call message is any messages which is not following within the context of a call and which does not form part of registration processing. These are always classified as either a request or a response.)

You can rate limit all in-call and out-of-call messages.

This includes in-call messages at all scopes, as normal. For example:

- Configuration at the “per-call” scope allows you to limit the rate at which an endpoint sends messages within a call.
- Configuration at the “dst-adjacency” scope allows you to limit the total rate of in-call messages sent out of an adjacency within all of the calls using that adjacency. (This could ensure that the load out of an adjacency never exceeds that which the attached network entity can cope with.)

The following messages are not rate-limited:

- SIP INVITE requests: 200 responses and ACK messages
- SIP PRACK messages and response
- SIP BYE messages and responses
- Any SIP message with non-duplicate SDP on
- For H.323 calls: Q.931 SETUP, Q.931 CONNECT and Q.931 RELEASE messages.

You can place restrictions on the rate at which out-of-call messages are processed. Configuration is permitted at all scopes except per-call scope (because this scope does not exist for out-of-call messages).

The Cisco Unified Border Element (SP Edition) will gracefully reject in-call messages when the rate exceeds that specified in the CAC. When an in-call message is not processed, the Cisco Unified Border Element (SP Edition) does the following:

- For SIP messages, Cisco Unified Border Element (SP Edition) rejects the message gracefully wherever possible. The rejection is sent back to the sending endpoint, so the call is likely to survive.
- For H.323 messages, Cisco Unified Border Element (SP Edition) drops the message because they usually cannot be gracefully rejected. This is likely to be disruptive for the call.

The Cisco Unified Border Element (SP Edition) gracefully rejects out-of-call messages when the rate exceeds that specified in CAC.

All rate limits must be protocol stack independent; limits must police SIP and H323messages.

In addition to configuring blacklists based on a number of CAC policy failures, you can now allow blacklists to be applied to endpoints that send in-call or out-of-call messages at a high rate.

Multiple CAC Averaging Periods

The user can apply different rate limits over a different averaging period by configuring a second set of rate-limiting CAC criteria. The user is able to do the following:

- Set the averaging period for the secondary rate calculation.
- Set the maximum number of new calls per minute for the secondary rate calculation, if a limit is required.
- Set the maximum number of endpoint registrations per minute for the secondary rate calculation, if a limit is required.
- Set the maximum number of in-call messages to be processed per minute for the secondary rate calculation, if a limit is required.
- Set the maximum number of out-of-call messages to be processed per minute for the secondary rate calculation, if a limit is required.

The user can configure two sets of SBC policies together that have rate-limiting criteria. The CAC rejects an event if it breaks any of the configured limits.

Subscriber Policy

A user can subscribe multiple endpoints to the network to allow them to make calls. A subscriber is one of those endpoints. In a particular network, you might want to limit each subscriber to no more than a specific number of simultaneous calls. The Subscriber Policy feature allows you to limit each subscriber to a specific number of simultaneous calls.

This feature provides the ability to configure the CAC limits. For example, you can configure the maximum number of concurrent calls, the maximum number of registrations, or the maximum call rate at different scopes, such as subscriber, subscriber category, and subscriber category prefix.

You can configure CAC tables:

- To associate a subscriber with a subscriber category. Call events between that subscriber and the core network are also associated with that same subscriber category.
- To match on a subscriber category or on a subscriber category prefix (the first n bits of the subscriber category), and then set limits when matched. The subscriber category prefix specifies the length of prefix to match. If specified, then only the first n bits of each of the call's subscriber categories is checked for a match.
- To set limits per subscriber category.
- To set limits per subscriber.

Note that when a subscriber scope is enabled, the SBC tracks an additional group of ALL "non-subscribers." The non-subscriber group is matched if the call is from a non-subscriber. Limits set in the subscriber scope apply to this non-subscriber group.

Privacy Service

The SBC provides the privacy service to ensure that requests for anonymity, as requested by a user during signaling, can be dynamically acted upon to ensure that the user's anonymity is maintained when the user leaves a trusted network. A user can request various levels of anonymity, with the privacy service removing the information that a user wants to withhold. The SBC can be configured such that individual adjacencies can be marked as trusted, untrusted, or configured in order to apply the privacy service. The privacy service is applied in a CAC policy set.

In addition to this, the SBC can edit—override or modify—a user's request for privacy when forwarding the privacy request. For example, a user can request identity of self to be withheld, but by editing the privacy request, the identity can be provided.

A user can also provide indications of anonymity in the display and presentation number. During number analysis, these calls can be detected and different analysis trees be used to progress the call.

The Privacy Service feature provides the following functions:

- Apply a privacy service based on information provided by a user when leaving a trusted domain.
- Edit a privacy service on request from a user and perform functions such as pass, strip, insert, and replace indications.
- Declare configurable trust boundaries.
- Detect calls in number analysis where the source is anonymous.
- Standard SIP header rewriting is performed by the SBC to cover the additional requirements specified in the SIP privacy header:
  - The Call ID, Server, and Contact headers are rewritten to hide the endpoint's identity.
  - Any Via headers are cached and replaced on the message with a single header identifying the SBC.

Both SIP and H323 adjacencies allow the configuration of the trusted and untrusted statuses.

For information about configuring the Privacy Service feature, see the ?$paranum>Configuring Privacy Service? section on page 7-126.

Session Initiation Protocol

In the context of SIP, a user indicates the levels of privacy that should be applied using the Privacy header. If the SBC cannot recognize any of the tokens present in the header, the message is rejected with a 433 Anonymity Disallowed response. Similarly, a response containing a critical privacy request that cannot be met is converted to a 433 failure response for an in-call message. For an out-of-dialog message, the response is dropped to ensure that no private information gets leaked accidently.

If this is an in-call message that does not contain a privacy header, the privacy requirements are assumed to be the same as those specified in the last privacy header from the side of the call. However, if the SBC reroutes a call locally, for example, a SIP 3xx redirect response, it discards the previously learnt privacy requirements on the side of the call that has been rerouted.

The following events occur when privacy services are applied to a request or response:

- When the privacy service based on a user, *Privacy: user*, is applied to a request or response, the Reply-To, Call-Info, User-Agent, Organization, Subject, In-Reply-To, Warning, and Server headers are stripped from the message.

  Also, when the privacy service based on a user, *Privacy: user*, is applied to a request, the URI in the From header is rewritten to anonymous@anonymous.invalid. The original URI is stored for replacement on responses. The display name in the From header is removed, and any further header manipulation rules that are configured as part of the user ID privacy are applied to the message.
- When the privacy service based on ID value, *Privacy: id*, is applied to a request or a response, the P-Preferred-ID, P-Asserted-Identity, and Remote-Party-Id headers are stripped from the message.
- When the privacy service based on session privacy, *Privacy: session*, is applied to a request or a response, media bypass is disallowed. However, if the session privacy is critical, and it is too late to disable media bypass, the call is torn down.
- When the privacy service based on header privacy, *Privacy:header*, is applied to a request or response, Record-Route or Route headers, if any, are removed and stored. They are restored on the responses within the dialog. If any further header manipulation rules are configured, they are applied to the message.The SBC strips the Privacy header from the ongoing message and removes the *privacy* option-tag, if any, from the Proxy-Require header.

Users can dynamically request for privacy service. This service can be applied by inserting *Privacy: header* based on RFC 3323 and RFC 3325.

Privacy Service on SIP Requests

Table 7-8 lists the behavior of the privacy service when it is applied on SIP requests, and *Privacy: header* is present to indicate the appropriate level of privacy to be applied.

Table 7-8 Privacy Service on SIP Requests

Header Name	None	User	Header	ID
From	—	Set to anonymous value: anonymous@anonymous.invalid	—	
Contact	—	—	Rewritten	—
Reply-to	—	Stripped	—	—
Via	Stripped	Stripped	Stripped	Stripped
Call-Info	—	Stripped	—	—

Table 7-8 Privacy Service on SIP Requests (continued)

User-Agent	—	Stripped	—	—
Organization	—	Stripped	—	—
Server	—	—	—	—
Subject	—	Stripped	—	—
Call-ID	Rewritten	Rewritten	Rewritten	Rewritten
In-Reply-To	—	Stripped	—	—
Warning	—	—	—	—
P-Asserted-Identity	—	—	—	Stripped
P-Preferred-Identity	—	—	—	Stripped
Remote-Party-ID	—	—	—	Stripped
Record-Route	—	—	Stripped	—

Privacy Service on SIP Responses

Table 7-9 lists the behavior of the privacy service when it is applied on SIP responses.

Table 7-9 Privacy Service on SIP Responses

Header Name	None	User	Header	ID
From	—	—	—	—
Contact	—	—	Rewritten	—
Reply-to	—	Stripped	—	—
Via	—	—	—	—
Call-Info	—	Stripped	—	—
User-Agent	—	Stripped	—	—
Organization	—	Stripped	—	—
Server	—	Stripped	—	—
Subject	—	—	—	—
Call-ID	Rewritten	Rewritten	Rewritten	Rewritten
In-Reply-To	—	—	—	—
Warning	—	Stripped	—	—
P-Asserted-Identity	—	—	—	Stripped
P-Preferred-Identity	—	—	—	Stripped
Remote-Party-ID	—	—	—	Stripped
Record-Route	—	—	Stripped	—

Privacy Service on H.323

The SBC treats the following H.323 protocol events as requests for the privacy service:

- On a Q.931 Setup, the caller address presentation restriction is requested if the Q.931 callingPartyNumber is present, and contains a presentationIndicator set to 3, presentation restricted, or, the H.225 presentationIndicator is present and set to presentationRestricted.
- On a Q.931 Connect, callee address presentation restriction is requested if the Q.931 connectedNumber is present, and contains a presentationIndicator set to 3, presentation restricted, or, the H.225 presentationIndicator is present and set to presentationRestricted.

When there is a conflict between the two presentationIndicators, the value in the Q.931 callingPartyNumber, the connectedNumber, takes precedence.

H.323 to SIP

A presentation restriction indication that is received for the callingPartyNumber or connectedNumber elements in an H.323 message is considered a request for *header;id;critical* privacy when being translated to a SIP privacy request.

If the presentation restriction is requested by the H.323 side, the URI in the From header is rewritten with anonymous@anonymous.invalid whether or not the SBC is acting as a privacy service.

When interworking with the SIP, the privacy service is always applied.

SIP to H.323

A SIP-signaled request for *id* or *header* privacy is translated into an H.323 presentation restriction on outgoing addresses, if any. All the other SIP privacy tokens are ignored.

Message, Policy, and Subscriber Statistics

From Cisco IOS XE Release 3.3S, enhancements have been made to the following statistics:

- Call Statistics, page 7-50
- CAC Statistics, page 7-55
- Subscriber Statistics, page 7-58

Call Statistics

The call-related statistics have been enhanced to include the following two features:

- Number of calls completed during a period—The summary periods of the call-related statistics includes the total number of calls that have been completed. A call is completed because of a signaling message received from an upstream or a downstream device. For SIP calls, a call is completed when a participating endpoint sends a BYE message. For H.323 calls, a call is completed when a participating endpoint sends a ReleaseComplete message with causeValue of 16, which indicates a *normal call clearing*.
- Running total for the calls statistics—The summary periods of the call-related statistics includes a *current-indefinite* time value. This time value provides the call-statistics for a period since the value has been last reset. Initially, the *current-indefinite* time value displays the statistics for the time

period since the router was booted. After the value has been reset, it displays the statistics for the time when the last reset was done. The time is in a UTC format where the year, month, day and time is displayed.

The following commands are used for displaying the call-related statistics and resetting the call-related statistics:

- The **show sbc** *sbc-name* **sbe call-stats** {**all** | **global** | **per-adjacency** *adjacency-name* | **src-account** *name* | **dst-account** *name* | **src-adjacency** *name* | **dst-adjacency** *name*} *period* command—Lists the statistics pertaining to all the calls on a SBE for a particular period, such as *currentindefinite*.
- The **clear sbc** *sbc-name* **sbe call-stats** [**all** | **dst-account** *account-name* | **dst-adjacency** *adjacency-name* | **global** | **src-account** *account-name* | **src-adjacency** *adjacency-name* | **per-adjacency** *adjacency-name*] [**all** | **current-indefinite**] command—Clears the call statistics on a SBE by the current-indefinite period.

The following example shows how the **show sbc sbe call-stats all adj1 currentindefinite** command displays statistic pertaining to all calls on the SBE for the current-indefinite period:

```
Router# show sbc SBC2 sbe call-stats all currentindefinite

statistics for the current indefinite for source adjacency phone1
Call count totals:
  Total call attempts =                                          1
  Total active calls =                                           0
  Total active IPv6 calls =                                      0
  Total activating calls =                                       0
  Total de-activating calls =                                    0
  Total active emergency calls =                                 0
  Total active e2 emergency calls =                              0
  Total IMS rx active calls =                                    0
  Total IMS rx call renegotiation attempts =                     0
  Total SRTP-RTP interworked calls =                             0
  Total active calls not using SRTP =                            0
  Total active transcoded calls =                                0
  Total active transrated calls =                                0
  Total calls completed =                                        1

General call failure counters:
  Total call setup failures =                                    0
  Total active call failures =                                   0
  Total failed call attempts =                                   0
  Total failed calls due to update failure =                     0
  Total failed calls due to resource failure =                   0
  Total failed calls due to congestion =                         0
  Total failed calls due to media failure =                      0
  Total failed calls due to signaling failure =                  0
  Total failed calls due to IMS rx setup failure =               0
  Total failed calls due to IMS rx renegotiation failure =       0
  Total failed calls due to RTP disallowed on call leg =         0
  Total failed calls due to SRTP disallowed on call leg =        0

Policy control failures:
  Call setups failed due to NA =                                 0
  Call setups failed due to RTG =                                0
  Call setups failed due to CAC =                                0
  CAC fails due to number of calls limit =                       0
  CAC fails due to call rate limit =                             0
  CAC fails due to bandwidth limit =                             0
  CAC fails due to number of media channels limit =              0
  CAC fails due to number of media update limit =                0
  CAC message drops due to mid call message rate limit =         0
  CAC message drops due to out of call message rate limit =      0
```

```
Stats Reset Timestamp:
  Timestamp when stats for this summary period were reset =      2011/03/07 03:27:36
statistics for the current indefinite for destination adjacency phone2
Call count totals:
  Total call attempts =                                                 1
  Total active calls =                                                  0
  Total active IPv6 calls =                                             0
  Total activating calls =                                              0
  Total de-activating calls =                                           0
  Total active emergency calls =                                        0
  Total active e2 emergency calls =                                     0
  Total IMS rx active calls =                                           0
  Total IMS rx call renegotiation attempts =                            0
  Total SRTP-RTP interworked calls =                                    0
  Total active calls not using SRTP =                                   0
  Total active transcoded calls =                                       0
  Total active transrated calls =                                       0
  Total calls completed =                                               1

General call failure counters:
  Total call setup failures =                                           0
  Total active call failures =                                          0
  Total failed call attempts =                                          0
  Total failed calls due to update failure =                            0
  Total failed calls due to resource failure =                          0
  Total failed calls due to congestion =                                0
  Total failed calls due to media failure =                             0
  Total failed calls due to signaling failure =                         0
  Total failed calls due to IMS rx setup failure =                      0
  Total failed calls due to IMS rx renegotiation failure =              0
  Total failed calls due to RTP disallowed on call leg =                0
  Total failed calls due to SRTP disallowed on call leg =               0

Policy control failures:
  Call setups failed due to NA =                                        0
  Call setups failed due to RTG =                                       0
  Call setups failed due to CAC =                                       0
  CAC fails due to number of calls limit =                              0
  CAC fails due to call rate limit =                                    0
  CAC fails due to bandwidth limit =                                    0
  CAC fails due to number of media channels limit =                     0
  CAC fails due to number of media update limit =                       0
  CAC message drops due to mid call message rate limit =                0
  CAC message drops due to out of call message rate limit =             0

Stats Reset Timestamp:
  Timestamp when stats for this summary period were reset =      2011/03/07 03:27:36
statistics for the current indefinite for source account source1
Call count totals:
  Total call attempts =                                                 1
  Total active calls =                                                  0
  Total active IPv6 calls =                                             0
  Total activating calls =                                              0
  Total de-activating calls =                                           0
  Total active emergency calls =                                        0
  Total active e2 emergency calls =                                     0
  Total IMS rx active calls =                                           0
  Total IMS rx call renegotiation attempts =                            0
  Total SRTP-RTP interworked calls =                                    0
  Total active calls not using SRTP =                                   0
  Total active transcoded calls =                                       0
  Total active transrated calls =                                       0
  Total calls completed =                                               1
```

```
General call failure counters:
  Total call setup failures =                                    0
  Total active call failures =                                   0
  Total failed call attempts =                                   0
  Total failed calls due to update failure =                     0
  Total failed calls due to resource failure =                   0
  Total failed calls due to congestion =                         0
  Total failed calls due to media failure =                      0
  Total failed calls due to signaling failure =                  0
  Total failed calls due to IMS rx setup failure =               0
  Total failed calls due to IMS rx renegotiation failure =       0
  Total failed calls due to RTP disallowed on call leg =         0
  Total failed calls due to SRTP disallowed on call leg =        0

Policy control failures:
  Call setups failed due to NA =                                 0
  Call setups failed due to RTG =                                0
  Call setups failed due to CAC =                                0
  CAC fails due to number of calls limit =                       0
  CAC fails due to call rate limit =                             0
  CAC fails due to bandwidth limit =                             0
  CAC fails due to number of media channels limit =              0
  CAC fails due to number of media update limit =                0
  CAC message drops due to mid call message rate limit =         0
  CAC message drops due to out of call message rate limit =      0

Stats Reset Timestamp:
  Timestamp when stats for this summary period were reset =      2011/03/07 03:27:36
statistics for the current indefinite for destination account dest1
Call count totals:
  Total call attempts =                                          1
  Total active calls =                                           0
  Total active IPv6 calls =                                      0
  Total activating calls =                                       0
  Total de-activating calls =                                    0
  Total active emergency calls =                                 0
  Total active e2 emergency calls =                              0
  Total IMS rx active calls =                                    0
  Total IMS rx call renegotiation attempts =                     0
  Total SRTP-RTP interworked calls =                             0
  Total active calls not using SRTP =                            0
  Total active transcoded calls =                                0
  Total active transrated calls =                                0
  Total calls completed =                                        1

General call failure counters:
  Total call setup failures =                                    0
  Total active call failures =                                   0
  Total failed call attempts =                                   0
  Total failed calls due to update failure =                     0
  Total failed calls due to resource failure =                   0
  Total failed calls due to congestion =                         0
  Total failed calls due to media failure =                      0
  Total failed calls due to signaling failure =                  0
  Total failed calls due to IMS rx setup failure =               0
  Total failed calls due to IMS rx renegotiation failure =       0
  Total failed calls due to RTP disallowed on call leg =         0
  Total failed calls due to SRTP disallowed on call leg =        0

Policy control failures:
  Call setups failed due to NA =                                 0
  Call setups failed due to RTG =                                0
  Call setups failed due to CAC =                                0
```

```
  CAC fails due to number of calls limit =                          0
  CAC fails due to call rate limit =                                0
  CAC fails due to bandwidth limit =                                0
  CAC fails due to number of media channels limit =                 0
  CAC fails due to number of media update limit =                   0
  CAC message drops due to mid call message rate limit =            0
  CAC message drops due to out of call message rate limit =         0

Stats Reset Timestamp:
  Timestamp when stats for this summary period were reset =      2011/03/07 03:27:36
statistics for the current indefinite for global counters
Call count totals:
  Total call attempts =                                             1
  Total active calls =                                              0
  Total active IPv6 calls =                                         0
  Total activating calls =                                          0
  Total de-activating calls =                                       0
  Total active emergency calls =                                    0
  Total active e2 emergency calls =                                 0
  Total IMS rx active calls =                                       0
  Total IMS rx call renegotiation attempts =                        0
  Total SRTP-RTP interworked calls =                                0
  Total active calls not using SRTP =                               0
  Total active transcoded calls =                                   0
  Total active transrated calls =                                   0
  Total calls completed =                                           1

General call failure counters:
  Total call setup failures =                                       0
  Total active call failures =                                      0
  Total failed call attempts =                                      0
  Total failed calls due to update failure =                        0
  Total failed calls due to resource failure =                      0
  Total failed calls due to congestion =                            0
  Total failed calls due to media failure =                         0
  Total failed calls due to signaling failure =                     0
  Total failed calls due to IMS rx setup failure =                  0
  Total failed calls due to IMS rx renegotiation failure =          0
  Total failed calls due to RTP disallowed on call leg =            0
  Total failed calls due to SRTP disallowed on call leg =           0

Policy control failures:
  Call setups failed due to NA =                                    0
  Call setups failed due to RTG =                                   0
  Call setups failed due to CAC =                                   0
  CAC fails due to number of calls limit =                          0
  CAC fails due to call rate limit =                                0
  CAC fails due to bandwidth limit =                                0
  CAC fails due to number of media channels limit =                 0
  CAC fails due to number of media update limit =                   0
  CAC message drops due to mid call message rate limit =            0
  CAC message drops due to out of call message rate limit =         0

Stats Reset Timestamp:
  Timestamp when stats for this summary period were reset =      2011/03/07 03:27:36
statistics for the current indefinite for adjacency phone1

Stats Reset Timestamp:
  Timestamp when stats for this summary period were reset =      2011/03/07 03:27:36
Current count of Media Packets Lost =                               0
Current count of Media Packets Dropped =                            0
Current count of Media Packets Sent =                             236
Current count of Media Packets Received =                         236
Current count of RTCP Packets Sent =                                0
```

```
Current count of RTCP Packets Received =                              0
Average Call Duration =                                           22004
Average of the Answer Seizure Ratio =                                 0
Average of the Round Trip Delay =                                     0 ms
Average of the locally calculated jitter =                            0 ms
Average of the remotely calculated jitter =                           0 ms
Average of the received media dropped per thousand pkts =             0
Average of the sent media lost per thousand pkts =                    0
statistics for the current indefinite for adjacency phone2

Stats Reset Timestamp:
  Timestamp when stats for this summary period were reset =      2011/03/07 03:27:36
Current count of Media Packets Lost =                                 0
Current count of Media Packets Dropped =                              0
Current count of Media Packets Sent =                               236
Current count of Media Packets Received =                           236
Current count of RTCP Packets Sent =                                  0
Current count of RTCP Packets Received =                              0
Average Call Duration =                                           22004
Average of the Answer Seizure Ratio =                              1000
Average of the Round Trip Delay =                                     0 ms
Average of the locally calculated jitter =                            0 ms
Average of the remotely calculated jitter =                           0 ms
Average of the received media dropped per thousand pkts =             0
Average of the sent media lost per thousand pkts =                    0
```

CAC Statistics

The CAC-related statistics have been enhanced to include the rejection counts for the CAC policies that have been implemented but failed.

A limiting field is configured in the CAC policy table entries and the CAC policy fails when there is a breach of that limit.

The following commands are used for displaying and clearing the CAC policy sets:

- The **show sbc** *name* **sbe cac-policy-set** [*id* [**table** *name* [**entry** *id*]] | **global** [**table** *name* [**entry** *id*]]] [**detail**] command—Lists information pertaining to the rejection counts for the failed CAC policies.
- The **clear sbc** *sbc-name* **sbe cac-policy-set-stats** [**all** | **policy-set** *cac-policy-number*] command—Clears all CAC policy statistics or clears statistics pertaining to a specified CAC policy set.

The following example shows how the **show sbc sbe cac-policy-set table entry** command displays statistic pertaining to the rejection counts for the failed CAC policies:

```
Router# show sbc SBC2 sbe cac-policy-set 1 table table entry 1

SBC Service "SBC2"
CAC Averaging period 1: 60 sec
CAC Averaging period 2: 0 sec

CAC Policy Set 1
  Global policy set: Yes
  Description:
  First CAC table: table
  First CAC scope: global

  Table name: table
    Description:
    Table type: policy-set
    Total call setup failures (due to non-media limits): 0
```

```
Entry 1
CAC scope:
CAC scope prefix length: 0
Action: CAC complete
Number of call setup failures (due to non-media limits): 0
No. of registrations rejected (due to registration limits): 0

Max calls per scope:                                Unlimited
No. of events rejected due to Max Call Limit:       0

Max reg. per scope:                                 Unlimited
No. of events rejected due to Max Reg limit:        0

Max channels per scope:                             Unlimited
Max updates per scope:                              Unlimited
Max bandwidth per scope:                            Unlimited


                                                 Averaging-period 1      Averaging-period
2

Max call rate per scope:                         Unlimited               Unlimited
No. of events rejected due to Max call rate:     0                       0

Max reg. rate per scope:                         Unlimited               Unlimited
No. of events rejected due to Max reg rate:      0                       0

Max in-call message rate:                        Unlimited               Unlimited
No. of events rejected due to Max in-call rate:  0                       0

Max out-call message rate:                       Unlimited               Unlimited
No. of events rejected due to Max Out call rate: 0                       0


Timestamp when the rejection counts were last reset: 2011/03/07 04:38:24

Early media:             Allowed         Early media direction:   Both
Early media timeout:     None            Transcoder per scope:    Allowed
Callee Bandwidth-Field:  None            Caller Bandwidth-Field:  None
Media bypass:            Allowed         Asymmetric Payload Type: Not Set
Renegotiate Strategy:            Delta
SRTP Transport:                  Trusted-Only (by default)
Caller hold setting:             Standard
Callee hold setting:             Standard
Caller limited-privacy-service:  Never hide identity
Callee limited-privacy-service:  Never hide identity
Caller privacy-service:  Not set
Callee privacy-service:  Not set
Caller edit-privacy-request:  Not set
Callee edit-privacy-request:  Not set
Caller edit-privacy-request sip strip:  Not set
Callee edit-privacy-request sip strip:  Not set
Caller edit-privacy-request sip insert:  Not set
Callee edit-privacy-request sip insert:  Not set
Caller voice QoS profile:        Default
Callee voice QoS profile:        Default
Caller video QoS profile:        Default
Callee video QoS profile:        Default
Caller sig QoS profile:          Default
Callee sig QoS profile:          Default
Caller inbound SDP policy:       None
Callee inbound SDP policy:       None
Caller outbound SDP policy:      None
```

```
Callee outbound SDP policy:      None
SDP Media Profile         :      None
Caller Generic Stream     :      Default
Callee Generic Stream     :      Default
Caller media disabled:           None
Callee media disabled:           None
Caller unsignaled secure media:  Not Allowed
Callee unsignaled secure media:  Not Allowed
Caller response downgrade support:  No
Callee response downgrade support:  No
Caller retry rtp support:           No
Callee retry rtp support:           No
Resend sdp answer in 200ok:      No
Caller tel-event payload type:   Default
Callee tel-event payload type:   Default
Media flag:                      None
Restrict codecs to list:         Default
Restrict caller codecs to list:  Default
Restrict callee codecs to list:  Default
Codec preference list:           Default
Caller Codec profile:           None
Callee Codec profile:           None
Caller media caps list:          None
Callee media caps list:          None
TCS extra codec list:            None
Caller media-type:               Inherit (default)
Callee media-type:               Inherit (default)
Caller Media Bypass:             Inherit (default)
Callee Media Bypass:             Inherit (default)
Media Bypass Type:               Not set
Callee local transfer support:   Inherit (default)
Maximum Call Duration:           50
Caller SRTP support:             Inherit (default)
Callee SRTP support:             Inherit (default)
SRTP Interworking:               Inherit (default)
SRTP media Interworking:         Inherit (default)
Ims rx preliminary-aar:          Disabled(default)
Ims media-service:               None(default)
media bandwidth policing:        Inherit(default)
Billing filter:                  Inherit(default)
Caller ptime:                    None (default)
Callee ptime:                    None (default)
Caller codec variant conversion: Disabled (default)
Callee codec variant conversion: Disabled (default)
Caller inband DTMF mode:         Inherit(default)
Callee inband DTMF mode:         Inherit(default)
Caller Port Range Tag:           Inherit (default)
Callee Port Range Tag:           Inherit (default)
Session refresh renegotiation: Inherit(default)
```

Subscriber Statistics

Subscriber-related statistics have been introduced to enable you to view and analyze information pertaining to the subscribers who are using the SBC. The **show sbc sbe** command has been enhanced to display these statistics.

The following are the features of subscriber-related statistics:

- The statistics include the subscribers who are currently registered with the SBC, that is, the subscribers stored in the SBC subscriber database. Therefore, only subscribers registered through a non-IMS Access adjacency, P-CSCF Access adjacency, or IPsec P-CSCF Access adjacency are included.
- The statistics include the individual access-side subscribers who register through the SBC. The significance of this feature is that if the SBC rewrites the register, the address of record forwarded to the registrar may be the same for two different access-side address of records. These two addresses of record are recorded as two subscribers in the statistics even though the registrar may consider this as a single subscriber. For example, if both sip:12345@192.0.2.22 and sip:12345@203.0.113.36 register through the SBC and the SBC rewrites them to sip:12345@example.registrar.com, the registrar treats this entry as a single subscriber. However, the SBC counts this entry as two subscribers.
- The statistics represent the number of registered subscribers and not the number of registered contacts. When a subscriber registers multiple contacts, that subscriber will be counted only against the source adjacency of the first contact who is registered. For example, if a subscriber registers Contact1 on Adjacency1 and then registers Contact2 on Adjacency2, the subscriber is counted only once in the global count of subscribers. The same subscriber is not included in the count of subscribers on Adjacency2. This holds true even if Contact1 expires and Contact2 remains active.
- The statistics include delegate registrations performed by the SBC.
- The statistics are available both globally and per adjacency.
- In the IMS mode, the SBC may store multiple addresses of record for a single subscriber, for example, if the registrar returns P-Associated-URIs on a REGISTER response. In this scenario, only a single subscriber is included in the statistics and not the additional addresses of record.

Restrictions for the Subscriber Statistics Feature

The following are the restrictions for the subscriber statistics feature:

- The statistics are lost in the event of a failover.
- The statistics do not include the number of fast-registered subscribers.
- If subscribers register through the SBC on an Interconnection Border Control Function (IBCF) or non-IMS adjacency, the SBC does not track these subscribers and they are, therefore, not included in the statistics.

The following command displays subscriber-related statistics:

show sbc *sbc-name* **sbe subscriber-stats** {**all** | **dst-account** *name* | **dst-adjacency** *name* | **global** | **src-account** *name* | **src-adjacency** *name*} **[current15mins | current5mins | currentday | currenthour | currentindefinite | previous15mins | previous5mins | previousday | previoushour]**

The following command resets all call-related statistics:

clear sbc *sbc-name* **sbe call-stats [all | dst-account** *name* **| dst-adjacency** *name* **| global | src-account** *name* **| src-adjacency** *name*] **[all | currentindefinite]**

The following example shows how the **show sbc sbe subscriber-stats** command displays statistics pertaining to subscribers:

```
Router# show sbc mySbc sbe subscriber-stats global currentindefinite

Subscribe count totals:
Active subscribers           =  10
Subscriber high water mark   =  15
Subscriber low water mark    =  3

Stats Reset Timestamp:
  Timestamp when stats for this summary period were reset =      2011/01/25 23:26:03
```

Administrative Domains

Each administrative domain represents administrative relationships with other peer entities, and can direct the SBC to use a particular number analysis and routing policy, and/or CAC policy for calls to and from the adjacency. The administrative domain is specified in admin-domain field for both SIP adjacencies and H.323 adjacencies. Any adjacencies without an administrative domain use globally configured policies.

An administrative domain has the following features:

- An administrative domain can identify policy trees that can be used for inbound or outbound number analysis, or taking a routing decision. These trees have all the attributes and capabilities of the existing number analysis and routing policy trees. The administrative domain can also identify zero or more CAC policy trees that have all the attributes and capabilities of the existing CAC policy tree.
- Users can create multiple separate policy trees for inbound number analysis, outbound number analysis, routing, and CAC. Each policy tree can be assigned to zero or more administrative domains. The user can bring each policy tree into and out of service independently from the others.
- An administrative domain can be identified by a text-based string that conveys the identity and scope of the domain.
- A signaling event can be assigned to a global administrative domain as its source or destination domain, if the classification system fails to assign it to any other source or destination administrative domain.
- Users can also assign a signaling message to multiple source and destination administrative domains. Each administrative domain is given a priority when it is assigned to an event. As per the priority given, the SBC uses the policy tree from the set of administrative domains.
- The user can assign a policy tree to administrative domain to take a routing decision for a signaling event. The user can also assign a policy tree to an administrative domain for outbound number analysis for the destination administrative domain. Changes to the policy trees can be made independently of each other. A routing decision is taken based on the policy tree chosen for the source administrative domain, but the outbound number analysis is based on the policy tree chosen for the destination administrative domain.
- All the source and destination administrative domains selected for a signaling event is provided to Billing Manager on detection points relating to that event. The XML format includes the names of the source and destination administrative domains in the billing record for a given call.

Asymmetric Payload Types

In Real-Time Transport Protocol (RTP) sessions, each codec is assigned an ID or payload type that is included in the RTP header. These payload types allow an RTP session to carry multiple formats, which may be different, concurrently. Different payload types can be assigned to the same codec in an RTP session.

If a session uses different payload types for the same codec, the session is said to be using asymmetric payload types.

SIP, H.323, and H.248 support asymmetric payload types. A SIP session negotiates the asymmetric payload types in RFC3264 Offer and Answer messages, while H.323 session negotiates the asymmetric payload types in the following messages:

- Fast Start request and response
- Open logical channel (OLC) and Open Logical Channel Acknowledgement
- Terminal Capabilities Set (applicable only to telephone-event codec)

The SBC enables seamless pass-through of asymmetric payload types in both signaling relay and media relay. Hence, an SBC can be used between two endpoints that use asymmetric payload types, without affecting the normal operations of the endpoints.

Asymmetric payload types are meant for only pass-through, and not for interworking. The SBC is not required to translate between asymmetric payload types on one leg of a call and symmetric payload types on the other leg of a call.

Prior to Cisco IOS XE Release 3.1.0S, if a SIP peer requested an asymmetric payload type, the SBC removed the codec that used the Asymmetric payload types. If no codecs were left, the entire call was torn down, as shown in Figure 7-8. From Cisco IOS XE Release 3.1.0S, the scenario illustrated in Figure 7-8 results in a successful call.

Figure 7-8 Asymmetric Payload Types —Call Teardown Scenarios

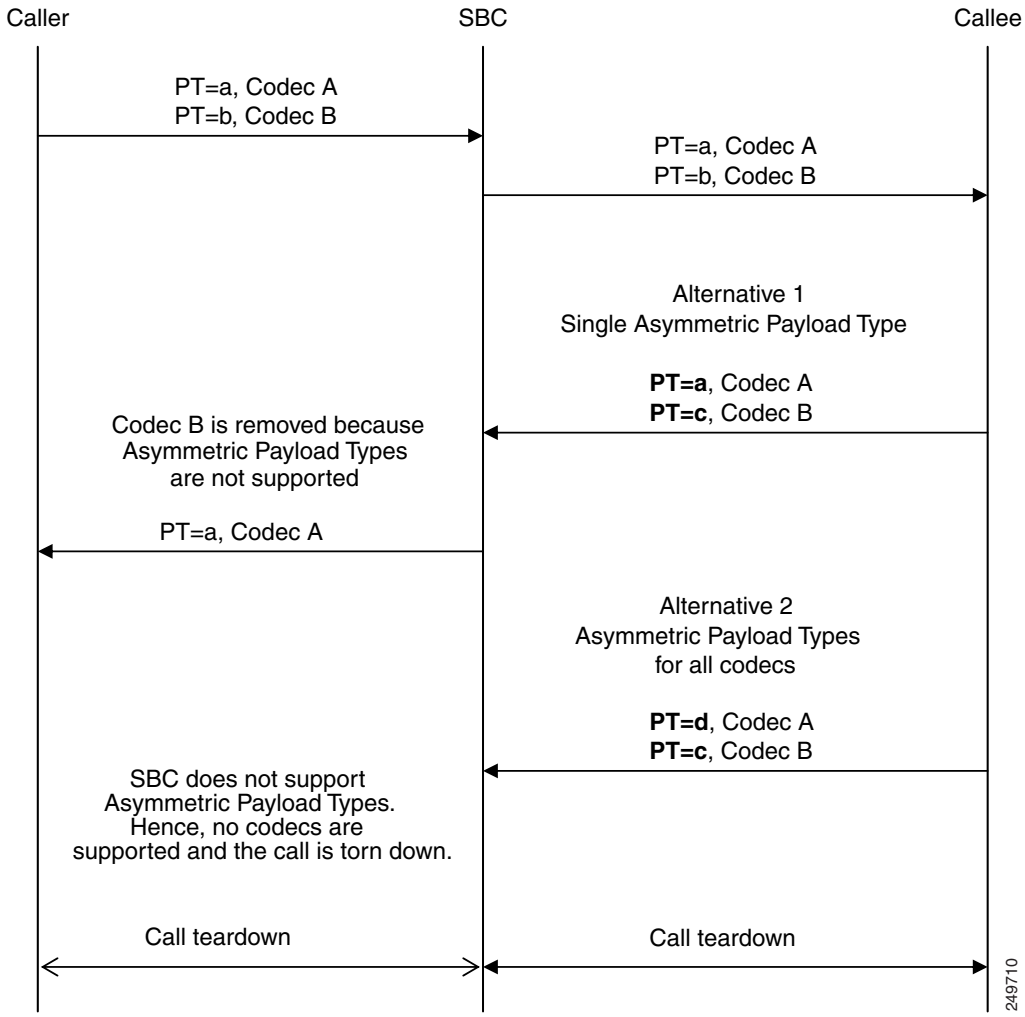
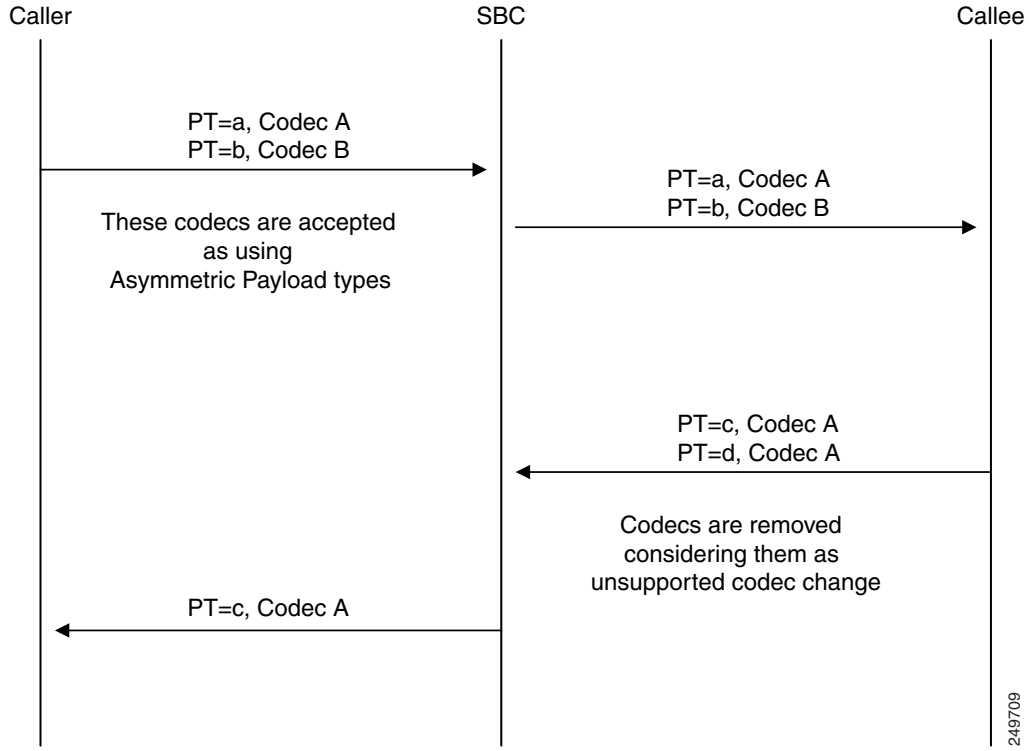


Figure 7-9 shows a scenario where two codecs are present in an Offer, but only one is present—but twice—in the Answer. In this scenario, a combination of Asymmetric payload types and a changed codec is present.

Figure 7-9 Asymmetric Payload Types—Two Codecs in Offer and One Codec in Answer



In the example illustrated in Figure 7-9, the SBC matches the Answer to the Offer, mapping the first codecs together as using Asymmetric payload types, and then discards the second set of codecs as an unsupported codec change.

Signaling

This section describes how the Asymmetric payload types feature works in the following scenarios:

- SIP-RFC3264 Offer-Answer
- H323-H245
- H.323-SIP Interworking
- Media Programming

SIP-RFC3264 Offer-Answer

This section provides details on how Asymmetric payload types works in a SIP-RFC3264 Offer-Answer scenario.

Asymmetric Payload Types:

- Communicate payload type reassignment from Answer onwards to the offerer.
- Communicate to the MEDIA asymmetric payload type bindings negotiated by RFC3264.
- Log an event when they detect an answer that changes corresponding payload type in a media relay call.

When signaling originates a codec in an offer—either the telephone-event codec in dual tone multifrequency (DTMF) interworking, or a transcoder codec in transcoding—signaling accepts a change of payload type on the answer.

Refer to RFC3264 for more details on how payload type bindings are assigned by SDP rtpmap line.

H323-H245

For H323 calls, asymmetric payload types support is available only for the telephone-event codec.

The Asymmetric Payload Type feature affects only the processing of H.245 Terminal Capability Set, in particular, the receiveRTPAudioTelephonyEventCapability, which signals the RFC2833 telephone-event payload type. This feature:

- Communicates to MEDIA the asymmetric telephone-event payload type bindings negotiated by the H.245 Terminal Capability Set.
- Generates logs when it detects a different telephone-event payload type in each direction.

H.323-SIP Interworking

For H.323-SIP interworking, asymmetric payload types support is available only for the telephone-event codec. The telephone-event payload type received in an H.245 Terminal Capability Set message is communicated onwards in an RFC3264 Offer or Answer message and vice versa.

Although H.323 does not support Asymmetric payload types for any codec other than telephone-event, the same restriction does not apply to a SIP. Hence, a SIP peer might attempt to change the payload type on a flow as part of a SIP-H.323 interworking call. If the payload type is changed, a high-severity Problem Determination log is created, and the call is discarded.

Media Programming

Signaling uses standard H.248 signaling to program asymmetric payload type streams. During the transitions between a Symmetric and Asymmetric payload type bindings, media addresses or ports are not reallocated.

Billing

The Asymmetric Payload Types feature provides the following information for billing:

- Asymmetric payload types that are in use for a given media relay call.
- Codecs that are bound to each payload type.

SIP-SIP Calls

SIP calls indicate Asymmetric payload types by indicating differing payload types in an answer to the previous offer. the SBC will then act upon these Asymmetric payload types.

See the ?$paranum>Example: Allowing Asymmetric Payload Types? section on page 7-164 for examples of SIP/SIP configuration and Offer-Answer messages.

Configuring Asymmetric Payload Types

You can configure the SBC to allow or block Asymmetric payload types for each call. By default, asymmetric payload types are allowed on calls.

Use the **[no] payload-type asymmetric {allowed | disallowed}** command to specify whether to allow or disallow asymmetric payload types.

Performing ISSU for Asymmetric Payload Types

When performing ISSU to upgrade to Cisco IOS XE Release 3.1.0S, a call requesting Asymmetric payload types from an active SBC with a release prior to Cisco IOS XE Release 3.1.0S is replicated to a standby with Cisco IOS XE Release 3.1.0S that supports Asymmetric payload types as if Symmetric Payload Types are being used. The media may not flow correctly on the primary or the backup after the failover.

If a call is currently using Symmetric payload types on an active SBC that does not support Asymmetric payload types, during attempts to renegotiate using Asymmetric payload types, one of the following occurs:

- If the Media, media forwarding component, or the Media Gateway detect that the active SBC does not support Asymmetric payload types, then the change to the corresponding call may be rejected and the call will remain unchanged.
- If the Media, media forwarding component, or Media Gateway does not detect that the active SBC does not support Asymmetric payload types, the corresponding call may continue as if using Symmetric payload types, and this may result in media not flowing correctly.

If a call changes the payload type from Symmetric to Asymmetric, or vice versa:

- After a gate is defined as Asymmetric, it remains Asymmetric even if it ceases to use Asymmetric payload types as a result of a renegotiation.
- If a Symmetric gate is marked as Asymmetric, and the partner does not support Asymmetric payload types, the gate is no longer replicated. The gate is deleted from the backup partner.

How to Implement Policies

Cisco Unified Border Element (SP Edition) policies are configured and activated as described in the following sections:

- Configuring Number Analysis Tables
- Configuring Administrative Domain
- Configuring Default Call Policy Set
- Configuring Routing Tables
- Configuring Number Manipulation
- Configuring Hunting
- Configuring H.323 MultiARQ Hunting
- Configuring Call Admission Control Policy Sets, CAC Tables, and Global CAC Policy Sets
- Configuring Privacy Service
- Configuring Multiple SBC Media Bypass
- Configuring Common IP Address Media Bypass
- Activating a CAC Policy Set

Configuring Number Analysis Tables

This task configures a number analysis table. The types of number analysis configuration are described in the following sections:

- Configuring Number Validation
- Configuring Number Categorization
- Configuring Text Address Validation and Source Address Manipulation

Configuring Number Validation

This task configures number validation for a number analysis table.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **call-policy-set** *policy-set-id*
5. **first-inbound-na-table** *table-name*
6. **na-dst-prefix-table** *table-name*
7. **entry** *entry-id*
8. **match-prefix** *key*
9. **action** [**next-table** *goto-table-name* | **accept** | **reject**]
10. **category** *category-name*

11. **entry** *entry-id*
12. **edit** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*]
13. **edit-cic** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*]
14. **match-prefix** *key*
15. **action** [**next-table** *goto-table-name* | **accept** | **reject**]
16. **category** *category-name*
17. **entry** *entry-id*
18. **match-prefix** *key*
19. **action** [**next-table** *goto-table-name* | **accept** | **reject**]
20. **category** *category-name*
21. **exit**
22. **exit**
23. **end**
24. **show**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.

Step 2	**sbc** *sbc-name*
	
	Example:
	`Router(config)# sbc mySbc`
	`Router(config-sbc)#`
		Enters the SBC service mode.
		• *sbc-name*—Name of the SBC.

Step 3	**sbe**
	
	Example:
	`Router(config-sbc)# sbe`
	`Router(config-sbc-sbe)#`
		Enters the mode of an SBE entity within an SBC service.

Step 4	**call-policy-set** *policy-set-id*
	
	Example:
	`Router(config-sbc-sbe)# call-policy-set 1`
	`Router(config-sbc-sbe-rtgpolicy)#`
		Enters the mode of routing policy set configuration within an SBE entity, creating a new policy set, if necessary.

Step 5	**first-inbound-na-table** *table-name*
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy)#`
	`first-inbound-na-table hotel_table`
		Configures the name of the first policy table to process when performing the number analysis stage of policy.

	Command or Action	Purpose
Step 6	**na-dst-prefix-table** *table-name* **Example:** `Router(config-sbc-sbe-rtgpolicy)# na-dst-prefix-table hotel_table`	Enters the mode for configuring a number analysis table whose entries match the prefix (the first several digits) of the dialed number within the context of an SBE policy set. Commands for other number analysis tables: • **na-carrier-id-table**—This table requires additional commands **match-cic** and **edit-cic** (see below) • **na-dst-address-table** • **na-src-address-table** • **na-src-prefix-table** • **na-src-account-table** • **na-src-adjacency-table** • **na-carrier-id-table**
Step 7	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable)# entry 1`	Enters the mode for configuring an entry in a number analysis table, creating the entry, if necessary.
Step 8	**match-prefix** *key* | **match-cic** *cic* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# match-prefix XXX`	Configures the match value of an entry in the number analysis table. • The **match-prefix** *key* argument is a string used to match the prefix (the starting part) of the dialed number. • The **match-cic** *cic* argument is used with the **na-carrier-id-table** command and configures the match carrier ID code in a table whose entries match a carrier ID.
Step 9	**action [next-table** *goto-table-name* **| accept | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# action accept`	Configures the action of an entry in a number analysis table. Possible actions are: • Configure the name of the next number analysis table to process if the event matches this entry using the **next-table** keyword and the *goto-table-name* argument. • Configure the call to be accepted if it matches the entry in the table using the **accept** keyword. • Configure the call to be rejected if it matches the entry in the table using the **reject** keyword.
Step 10	**category** *category-name* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# category external`	Configures the category of an entry in the number analysis table.

	Command or Action	Purpose
Step 11	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# entry 2`	Enters the mode for configuring an entry in a number analysis table, creating the entry, if necessary.
Step 12	**edit** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*] **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# edit del-prefix 1`	Configures a dial-string manipulation action in a number analysis table. You are not allowed to do this if the table is part of the active policy set. The **no** version of the command deletes the edit action of the given entry in the routing table. The **edit** command can be set to the following values: • **del-prefix** *pd*—Delete prefix *pd*, where *pd* is a positive integer specifying a number of digits to delete from the front of the dialed string. • **del-suffix** *sd*—Delete suffix *sd*, where *sd* is a positive integer specifying a number of digits to delete from the end of the dialed string. • **add-prefix** *pa*—Add prefix *pa*, where *pa* is a string of digits to add to the front of the dialed string. • **replace** *ds*—Replace *ds*, where *ds* is a string of digits that replaces the dialed string. In the example to the left, the **edit** command sets entry 2 to delete 1 digit from the beginning of the dialed string in the number analysis table.

	Command or Action	Purpose

Step 13	**edit-cic** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*]
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy-natable-entry)# edit-cic del-prefix 1`
		Configures a carrier identification code (CIC) manipulation action in a number analysis table.
		You are not allowed to do this if the table is part of the active policy set.
		• **del-prefix** *pd*: A positive integer specifying a number of digits to delete from the front of the carrier ID string.
		• **del-suffix** *sd*: A positive integer specifying a number of digits to delete from the end of the carrier ID string.
		• **add-prefix** *pa*: A string of digits to add to the front of the carrier ID string.
		• **replace** *ds*: A string of digits to replace the carrier ID string with.
		The "edit-cic del-prefix 1" command sets entry 2 to delete the first digit of the carrier ID in the current number analysis table.
		You can remove the CIC or carrier ID from outbound messages by specifying a replacement string of 0000 or by specifying a prefix deletion length of 4.
		For example:
		`edit-cic del-prefix 4  OR`
		`edit-cic replace 0000`

Step 14	**match-prefix** *key* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# match-prefix 9XXX`	Configures the match value of an entry in the number analysis table. The *key* argument is a string used to match the start of the dialed number. The **no** version of the command destroys the match value.
Step 15	**action [next-table** *goto-table-name* **| accept | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# action accept`	Configures the action of an entry in a number analysis table. Possible actions are: • Configure the name of the next number analysis table to process if the event matches this entry using the **next-table** keyword and the *goto-table-name* argument. • Configure the call to be accepted if it matches the entry in the table using the **accept** keyword. • Configure the call to be rejected if it matches the entry in the table using the **reject** keyword.
Step 16	**category** *category-name* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# category external`	Configures the category of an entry in the number analysis table.

	Command or Action	Purpose
Step 17	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# entry 3`	Enters the mode for configuring an entry in a number analysis table, creating the entry, if necessary.
Step 18	**match-prefix** *key* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# match-prefix 8XXX`	Configures the match value of an entry in the number analysis table. The *key* argument is a string used to match the start of the dialed number.
Step 19	**action [next-table** *goto-table-name* **| accept | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# action accept`	Configures the action of an entry in a number analysis table. Possible actions are: • Configure the name of the next number analysis table to process if the event matches this entry using the **next-table** keyword and the *goto-table-name* argument. • Configure the call to be accepted if it matches the entry in the table using the **accept** keyword. • Configure the call to be rejected if it matches the entry in the table using the **reject** keyword.
Step 20	**category** *category-name* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# category bar`	Configures the category of an entry in the number analysis table.
Step 21	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# exit`	Exits from the **entry** mode to the **natable** mode.
Step 22	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable)# exit`	Exits from the **natable** mode to the **callpolicy** mode.
Step 23	**end** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable)# end`	Exits the callpolicy mode to Privileged EXEC mode.
Step 24	**show** **Example:** `Router(config-sbc-sbe-rtgpolicy)# show`	Displays the current configuration information.

Configuring Number Categorization

This task configures number categorization for a number analysis table.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **call-policy-set** *policy-set-id*
5. **first-inbound-na-table** *table-name*
6. **na-src-account-table** *table-name*
7. **entry** *entry-id*
8. **match-account** *key*
9. **action** [**next-table** *goto-table-name* | **accept** | **reject**]
10. **entry** *entry-id*
11. **match-account** *key*
12. **action [next-table** *goto-table-name* | **accept** | **reject**]
13. **entry** *entry-id*
14. **match-account** *key*
15. **action [next-table** *goto-table-name* | **accept** | **reject]**
16. **na-dst-prefix-table** *table-name*
17. **entry** *entry-id*
18. **match-prefix** *key*
19. **category** *category-name*
20. **action [next-table** *goto-table-name* | **accept** | **reject]**
21. **entry** *entry-id*
22. **match-prefix** *key*
23. **category** *category-name*
24. **action [next-table** *goto-table-name* | **accept** | **reject]**
25. **end**
26. **show**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.

Step 2	**sbc** *sbc-name*
	
	Example:
	`Router(config)# sbc mySbc`
	`Router(config-sbc)#`
		Enters the SBC service mode.
		• *sbc-name*—Name of the SBC.

Step 3	**sbe**
	
	Example:
	`Router(config-sbc)# sbe`
	`Router(config-sbc-sbe)#`
		Enters the mode of an SBE entity within an SBC service.

Step 4	**call-policy-set** *policy-set-id*
	
	Example:
	`Router(config-sbc-sbe)# call-policy-set 1`
	`Router(config-sbc-sbe-rtgpolicy)#`
		Enters the mode of routing policy set configuration within an SBE entity, creating a new policy set if necessary.

Step 5	**first-inbound-na-table** *table-name*
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy)#`
	`first-inbound-na-table check_account`
		Configures the name of the first policy table to process when performing the number analysis stage of policy.

Step 6	**na-src-account-table** *table-name*
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy)#`
	`na-src-account-table check_account`
	`Router(config-sbc-sbe-rtgpolicy-natable)#`
		Enters the mode for configuring a number analysis table within the context of an SBE policy set with the entries of the table matching the source account.

Step 7	**entry** *entry-id*
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy-natable)# entry 1`
	`Router(config-sbc-sbe-rtgpolicy-natable-entry)#`
		Enters the mode for configuring an entry in a number analysis table, creating the entry, if necessary.

Step 8	**match-account** *key* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# match-account hotel_foo`	Configures the match value of an entry in the number analysis table. The *key* argument is a string used to match the source account.

	Command or Action	Purpose
Step 9	**action [next-table** *goto-table-name* **| accept | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# action next-table hotel_dialing_plan`	Configures the action of an entry in a number analysis table. Possible actions are: • Configure the name of the next number analysis table to process if the event matches this entry using the **next-table** keyword and the *goto-table-name* argument. • Configure the call to be accepted if it matches the entry in the table using the **accept** keyword. • Configure the call to be rejected if it matches the entry in the table using the **reject** keyword.
Step 10	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# entry 2`	Enters the mode for configuring an entry in a number analysis table, creating the entry, if necessary.
Step 11	**match-account** *key* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# match-account hotel_bar`	Configures the match value of an entry in the number analysis table. The *key* argument is a string used to match the source account.
Step 12	**action [next-table** *goto-table-name* **| accept | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# action next-table hotel_dialing_plan`	Configures the action of an entry in a number analysis table. Possible actions are: • Configure the name of the next number analysis table to process if the event matches this entry using the **next-table** keyword and the *goto-table-name* argument. • Configure the call to be accepted if it matches the entry in the table using the **accept** keyword. • Configure the call to be rejected if it matches the entry in the table using the **reject** keyword.
Step 13	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# entry 3`	Enters the mode for configuring an entry in a number analysis table, creating the entry, if necessary.
Step 14	**match-account** *key* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# match-account internal`	Configures the match value of an entry in the number analysis table. The *key* argument is a string used to match the source account.

	Command or Action	Purpose
Step 15	**action [next-table** *goto-table-name* **| accept | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# action accept`	Configures the action of an entry in a number analysis table. Possible actions are: • Configure the name of the next number analysis table to process if the event matches this entry using the **next-table** keyword and the *goto-table-name* argument. • Configure the call to be accepted if it matches the entry in the table using the **accept** keyword. • Configure the call to be rejected if it matches the entry in the table using the **reject** keyword.
Step 16	**na-dst-prefix-table** *table-name* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# na-dst-prefix-table hotel_dialing_plan`	Enters the mode for configuring a number analysis table within the context of an SBE policy set with the entries of the table matching the start of the dialed number.
Step 17	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# entry 1`	Enters the mode for configuring an entry in a number analysis table, creating the entry, if necessary.
Step 18	**match-prefix** *key* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# match-prefix XXX`	Configures the match value of an entry in the number analysis table. The *key* argument is a string used to match the start of the dialed number.
Step 19	**category** *category-name* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# category internal_call`	Specifies the category of an entry in a number analysis table.
Step 20	**action [next-table** *goto-table-name* **| accept | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# action accept`	Configures the action of an entry in a number analysis table. Possible actions are: • Configure the name of the next number analysis table to process if the event matches this entry using the **next-table** keyword and the *goto-table-name* argument. • Configure the call to be accepted if it matches the entry in the table using the **accept** keyword. • Configure the call to be rejected if it matches the entry in the table using the **reject** keyword.
Step 21	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# entry 2`	Enters the mode for configuring an entry in a number analysis table, creating the entry, if necessary.

	Command or Action	Purpose
Step 22	**match-prefix** *key* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# match-prefix 9XXX`	Configures the match value of an entry in the number analysis table. The *key* argument is a string used to match the start of the dialed number.
Step 23	**category** *category-name* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# category external_call`	Specifies the category of an entry in a number analysis table.
Step 24	**action [next-table** *goto-table-name* **| accept | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# action accept`	Configures the action of an entry in a number analysis table. Possible actions are: • Configure the name of the next number analysis table to process if the event matches this entry using the **next-table** keyword and the *goto-table-name* argument. • Configure the call to be accepted if it matches the entry in the table using the **accept** keyword. • Configure the call to be rejected if it matches the entry in the table using the **reject** keyword.
Step 25	**end** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# end`	Exits from the **entry** mode and returns to Privileged EXEC mode.
Step 26	**show** **Example:** `Router(config-sbc-sbe-rtgpolicy)# show`	Displays the current configuration information.

Configuring Text Address Validation and Source Address Manipulation

This task shows how to configure text address validation and source address manipulation for a number analysis table.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **call-policy-set** *policy-set-id*
5. **first-inbound-na-table** *table-name*
6. **na-dst-address-table** *table-name*
7. **entry** *entry-id*
8. **action** [**next-table** *goto-table-name* | **accept** | **reject**]

9. **edit-src** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*]
10. **match-address** *key* [**regex** | **digits**]
11. **entry** *entry-id*
12. **action** [**next-table** *goto-table-name* | **accept** | **reject**]
13. **edit-src** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*]
14. **match-address** *key* [**regex** | **digits**]
15. **exit**
16. **exit**
17. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.

Step 2	**sbc** *sbc-name*
	
	Example:
	`Router(config)# sbc mySbc`
	`Router(config-sbc)#`
		Enters the SBC service mode.
		Use the *sbc-name* argument to define the name of the service.

Step 3	**sbe**
	
	Example:
	`Router(config-sbc)# sbe`
	`Router(config-sbc-sbe)#`
		Enters the SBE entity mode within an SBC service.

Step 4	**call-policy-set** *policy-set-id*
	
	Example:
	`Router(config-sbc-sbe)# call-policy-set 1`
	`Router(config-sbc-sbe-rtgpolicy)#`
		Enters the routing policy set configuration mode within an SBE entity, creating a new policy set, if necessary.

Step 5	**first-inbound-na-table** *table-name*
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy)#`
	`first-inbound-na-table hotel_table`
		Configures the name of the first policy table to be processed when performing the number analysis stage of the policy.

	Command or Action	Purpose
Step 6	**na-dst-address-table** *table-name* **Example:** `Router(config-sbc-sbe-rtgpolicy)# na-dst-address-table room_table`	Enters the number analysis table mode for configuring a number analysis table whose entries match the prefix (the first few digits) of the dialed number within the context of an SBE policy set. The commands for other number analysis tables are: • **na-carrier-id-table** (This table requires additional commands—**match-cic** and **edit-cic**) • **na-dst-address-table** • **na-src-address-table** • **na-src-prefix-table** • **na-src-account-table** • **na-src-adjacency-table** • **na-carrier-id-table**
Step 7	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable)# entry 1`	Enters the number analysis table entry mode for configuring an entry in a number analysis table, creating the entry, if necessary.
Step 8	**action [next-table** *goto-table-name* **| accept | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# action accept`	Configures the action of an entry in a number analysis table. Possible actions are: • Configure the name of the next number analysis table to be processed if the event matches this entry, using the **next-table** keyword and the *goto-table-name* argument. • Configure the call to be accepted if it matches the entry in the table, using the **accept** keyword. • Configure the call to be rejected if it matches the entry in the table, using the **reject** keyword.
Step 9	**edit-src** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*] **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# edit-src del-prefix 3`	Configures the source address manipulation action in the NA table. This cannot be done if a table is part of the active policy set. The **no** version of the command removes the match value. • **del-prefix** *pd*: A positive integer specifying the number of digits to be delete from the front of the carrier ID string. • **del-suffix** *sd*: A positive integer specifying the number of digits to be deleted from the end of the carrier ID string. • **add-prefix** *pa*: A string of digits to be added to the front of the carrier ID string. • **replace** *ds*: A string of digits to replace the carrier ID string with.

	Command or Action	Purpose
Step 10	**match-address** *key* [**regex** | **digits**] **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# match-address 123456 digits`	Configures the match value of an entry in an NA table. To create a routing table that routes on user name, use the existing rtg-dst-address-table or rtg-src-address-table, and include a textual value in the match-address field. The SBC skips number analysis and performs only routing when the SIP message contains a user name. The SBC decides that an address is a user name (as opposed to a phone number) if the address contains any character other than 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, plus, hyphen, period, open-round-bracket, and close-round-bracket. When the SBC has decided that an address is a user name, the *X* in the routing tables is treated not as a wildcard character, but as a literal *X*. For example, the match value of *X* matches the username *X*, but not *A*. **Note** A direct string comparison is not done by NA. To compare a fixed string, a regex without any regex meta-characters can be used.
Step 11	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# entry 2`	Enters the number analysis table entry mode for configuring an entry in a number analysis table, creating the entry, if necessary.
Step 12	**action [next-table** *goto-table-name* **| accept | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# action accept`	Configures the action of an entry in a number analysis table. Possible actions are: • Configure the name of the next number analysis table to be processed if the event matches this entry, using the **next-table** keyword and the *goto-table-name* argument. • Configure the call to be accepted if it matches the entry in the table, using the **accept** keyword. • Configure the call to be rejected if it matches the entry in the table, using the **reject** keyword.

	Command or Action	Purpose
Step 13	**edit-src** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*] **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# edit-src del-suffix 1`	Configures the source address manipulation action in the NA table. This cannot be done if the table is a part of the active policy set. The **no** version of the command destroys the match value. • **del-prefix** *pd*: A positive integer specifying the number of digits to be deleted from the front of the carrier ID string. • **del-suffix** *sd*: A positive integer specifying the number of digits to be deleted from the end of the carrier ID string. • **add-prefix** *pa*: A string of digits to be added to the front of the carrier ID string. • **replace** *ds*: A string of digits to be replaced the carrier ID string with.
Step 14	**match-address** *key* [**regex** | **digits**] **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# match-address ^.* regex`	Configures the match value of an entry in an NA table. To create a routing table that routes on user name, use the existing rtg-dst-address-table or rtg-src-address-table, and include a textual value in the match-address field. The SBC skips number analysis and performs only routing when the SIP message contains a user name. The SBC decides that an address is a user name (as opposed to a phone number) if the address contains any character other than 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, plus, hyphen, period, open-round-bracket, and close-round-bracket. When the SBC has decided that an address is a user name, the *X* in the routing tables is treated not as a wildcard character, but as a literal *X*. For example, the match value of *X* matches the username *X*, but not *A*. **Note** A direct string comparison is not done by NA. To compare a fixed string, a regex without any regex meta-characters can be used.
Step 15	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# exit`	Exits from the entry mode and enters the natable mode.
Step 16	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable)# exit`	Exits from the natable mode and enters the call policy mode.
Step 17	**end** **Example:** `Router(config-sbc-sbe-rtgpolicy)# end`	Exits the call policy mode and enters the Privileged EXEC mode.

Configuring Administrative Domain

This task configures an administrative domain.

Note The policy sets must be in a complete state before they are assigned to an administrative domain. A default call-policy-set must be configured before the administrative domain mode is entered. If an inbound NA set, a routing set, or an outbound NA set is undefined, the administrative domain uses the values defined within the default call-policy-set.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **admin-domain** *name*
5. **description** [*line*]
6. **call-policy-set** {**inbound-na** *number* | **outbound-na** *number* | **rtg** *number*} [**priority** *priority-value*]
7. **cac-policy-set** *number*
8. **exit**
9. **adjacency sip | h323** *adjacency-name*
10. **admin-domain** *name*
11. **end**
12. **show sbc** *sbc-name* **sbe admin-domain** [**adjacency**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mySbc`	Enters the mode of an SBC service. • *sbc-name*—Defines the name of the SBC service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.
Step 4	**admin-domain** *name* **Example:** `Router(config-sbc-sbe)# admin-domain Domain1`	Enters the mode of an administrative domain. • *name*—Defines the administrative domain name that can be of 30 characters maximum.

	Command or Action	Purpose
Step 5	**description** [*line*] **Example:** `Router(config-sbc-sbe-ad)# description This is a description of DOMAIN1`	Assigns a text description to the administrative domain. • *line*—Describes the administrative domain.

Step 6	**call-policy-set** {**inbound-na** *number* | **outbound-na** *number* | **rtg** *number*} [**priority** *priority-value*]
	
	Example:
	`Router(config-sbc-sbe-ad)# call-policy-set rtg 2 priority 1`
	`Router(config-sbc-sbe-ad)# call-policy-set inbound-na 2 priority 1`
	`Router(config-sbc-sbe-ad)# call-policy-set outbound-na 2 priority 1`
		Configures a single call-policy-set or separate call-policy-sets for routing, inbound number analysis, and outbound number analysis. The policy sets must be in a complete state before they can be assigned to the policy set of an administrative domain.
		Note Specifying an inbound NA, a routing, or an outbound NA policy set is optional. If the policy sets are undefined, the admin-domain uses the values defined within the default call policy set.
		• **inbound-na**—Specifies the inbound number analysis policy
		• **outbound-na**—Specifies the outbound number analysis policy
		• **rtg**—Specifies the routing policy
		• **priority**—Specifies the priority of a policy-set.
		• *number*—An unique identifier for the policy set. The value can range from 1 to 2147483647.
		• *priority-value*—The priority value ranging from 1 to 10 where 10 indicates the highest priority. By default, the priority is set to 10.
		Note Priority is required because more than one administrative domain can be specified on an adjacency. The SBC uses the policy-set with the highest priority.

Step 7	**cac-policy-set** *number* **Example:** `Router(config-sbc-sbe-ad)# cac-policy-set 2`	Configures the cac-policy-set in an administrative domain. Only one cac-policy-set can be specified. The policy sets must be in a complete state before they can be assigned to the policy set of an administrative domain. • *number*—An unique identifier for the policy set. The value can range from 1 to 2147483647.
Step 8	**exit** **Example:** `Router(config-sbc-sbe-ad)# exit`	Exits the administrative domain mode and enters the SBE mode.
Step 9	**adjacency sip** | **h323** *adjacency-name* **Example:** `Router(config-sbc-sbe)# adjacency sip sipadj`	Enters the mode of an SBE SIP or H.323 adjacency. • adjacency-name—Defines the name of the SIP or H.323 adjacency. **Note** The H323 adjacency must be unattached to add, delete, or modify the admin-domain command.

	Command or Action	Purpose
Step 10	**admin-domain** *name* **Example:** `Router(config-sbc-sbe-adj-sip)# admin-domain Domain1`	Configures an administrative domain on an adjacency. • *name*—Defines the administrative domain name that can be of 30 characters maximum.
Step 11	**end** **Example:** `Router(config-sbc-sbe-adj-sip)# end`	Exits the SBE SIP or H.323 adjacency mode and enters the Privilege exec mode.
Step 12	**show sbc** *sbc-name* **sbe admin-domain** [**adjacency**] **Example:** `Router# show sbc MySBC sbe admin-domain`	Displays details of administrative domains configured on the SBC. • *sbc-name*—Defines the name of the SBC service. • **adjacency**—Lists the administrative domains per adjacency.

The following example shows the output of the **show sbc sbe admin-domain** command:

```
Router# show sbc mySBC sbe admin-domain
SBC Service "mySBC"
Global cac-policy-set:             2
Default call-policy-set/priority:  1/6

                                    cac              call-policy-set/priority
Administrative Domain           policy-set    inbound-na       routing   outbound-na
------------------------------ ---------- ------------- ------------- -------------
DOMAIN1                                 2          2/1           2/1           2/1
```

The following example shows the output of the **show sbc sbe admin-domain adjacency** command:

```
Router# show sbc mySBC sbe admin-domain adjacency
SBC Service "mySBC"
Adjacency Name                 Type  State       Admin-domain
------------------------------ ----  ----------  ------------------------------
SIPP1A                         SIP   Attached    DOMAIN1
```

Configuring Default Call Policy Set

This task configures a call-policy-set and sets a priority for the SBC to determine the default policy set to use when the administrative domain is not present.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **call-policy-set** *policy-set-id*
5. **first-inbound-na-table** *word*
6. **first-outbound-na-table** *word*
7. **complete**

8. **exit**
9. **call-policy-set default** *policy-set-id* [**priority** *priority*]
10. **end**
11. **show sbc** *sbc-name* **sbe call-policy-set** [**default**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mySbc`	Enters the mode of an SBC service. • *sbc-name*—The name of the SBC service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.
Step 4	**call-policy-set** *policy-set-id* **Example:** `Router(config-sbc-sbe)# call-policy-set 25`	Creates a new call policy set and enters SBE routing policy configuration mode. • policy-set-id—The call policy set number that can range from 1 to 2147483647.
Step 5	**first-inbound-na-table** *word* **Example:** `Router(config-sbc-sbe-rtgpolicy)# first-inbound-na-table InTable`	Specifies the first inbound number analysis table. • *word*—Inbound number analysis table name. The table length can be of 30 characters maximum.
Step 6	**first-outbound-na-table** *word* **Example:** `Router(config-sbc-sbe-rtgpolicy)# first-outbound-na-table InTable`	Specifies the first outbound number analysis table. • *word*—Outbound number analysis table name. The table length can be of 30 characters maximum.
Step 7	**complete** **Example:** `Router(config-sbc-sbe-rtgpolicy)# complete`	Completes the call-policy set after committing the full set.
Step 8	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy)# exit`	Exits the SBE routing policy mode.

	Command or Action	Purpose
Step 9	**call-policy-set default** *policy-set-id* [**priority** *priority*] **Example:** `Router(config-sbc-sbe)# call-policy-set default 25 priority 1`	Assigns the default call-policy-set id when an administrative domain is not specified on the adjacency or the specified administrative domain does not exist. • *policy-set-id*—The call policy set number, ranging from 1 to 2147483647. The policy set must be in a complete state before it can be assigned as the default policy. • **priority**—Specifies the priority to determine which active call-policy-set to use. The SBC uses the policy set with the highest priority. • *priority*— The priority value ranging from 1 to 10 with 10 indicating highest priority. By default, priority is set to 6. **Note** A default call-policy-set must be configured before the user enters the administrative domain mode. If an inbound NA set, a routing set, or an outbound NA set is undefined, the administrative domain uses the values defined within the default call-policy-set.
Step 10	**end** **Example:** `Router(config-sbc-sbe)# end`	Exits the SBE mode and enters the Privilege exec mode.
Step 11	**show sbc** *sbc-name* **sbe call-policy-set** [**default**] **Example:** `Router# show sbc mySBC sbe call-policy-set`	Displays details of the call policy sets configured on the SBC. • *sbc-name*—Defines the name of the SBC service. • **default**—Lists the information pertaining to the default call policy set.

The following example shows the output of the **show sbc sbe call-policy-set** command:

```
Router# show sbc mySBC sbe call-policy-set

SBC Service "mySBC"

Policy set 1
  Default policy set           : Yes (priority 6)
  First inbound NA table       :
  First call routing table     : TAB1
  First reg routing table      : TAB2
  First outbound NA table      :

  Table Name                   : TAB1
    Class                      : Routing
    Table type                 : rtg-src-adj
    Total Call-policy Failures : 0 (0 *)
    Entry      Match Value          Destination Adjacency  Action
    -----      -----------          ---------------------  ------
    1          SIPP1A               SIPP1B                 Routing complete
    2          SIPP1B               SIPP1A                 Routing complete

  Table Name                   : TAB2
    Class                      : Routing
```

```
    Table type                 : rtg-src-adj
    Total Call-policy Failures : 0 (0 *)
    Entry      Match Value          Destination Adjacency  Action
    -----      -----------          ---------------------  ------
    1          SIPP1A               Registrar              Routing complete
    2          SIPP1B               Registrar              Routing complete

Policy set 2
  Default policy set           : No
  First inbound NA table       :
  First call routing table     : TAB1
  First reg routing table      : TAB2
  First outbound NA table      :

  Table Name                   : TAB1
    Class                      : Routing
    Table type                 : rtg-src-adj
    Total Call-policy Failures : 0 (0 *)
    Entry      Match Value          Destination Adjacency  Action
    -----      -----------          ---------------------  ------
    1          SIPP1A               SIPP1B                 Routing complete
    2          SIPP1B               SIPP1A                 Routing complete

  Table Name                   : TAB2
    Class                      : Routing
    Table type                 : rtg-src-adj
    Total Call-policy Failures : 0 (0 *)
    Entry      Match Value          Destination Adjacency  Action
    -----      -----------          ---------------------  ------
    1          SIPP1A               Registrar              Routing complete
    2          SIPP1B               Registrar              Routing complete

Policy set 25
  Default policy set           : No
  First inbound NA table       : ADMINTable
  First call routing table     :
  First reg routing table      :
  First outbound NA table      : OutTable

*  Numbers in brackets refer to a call being rejected by a routing or number analysis
table because there were no matching entries in the table. This is also included in the
total figure.
```

The following example shows the output of the **show sbc sbe call-policy-set default** command:

```
Router# show sbc mySBC sbe call-policy-set default

SBC Service "mySBC"

Policy set 1
  Default policy set           : Yes (priority 6)
  First inbound NA table       :
  First call routing table     : TAB1
  First reg routing table      : TAB2
  First outbound NA table      :

  Table Name                   : TAB1
    Class                      : Routing
    Table type                 : rtg-src-adj
    Total Call-policy Failures : 0 (0 *)
    Entry      Match Value          Destination Adjacency  Action
    -----      -----------          ---------------------  ------
    1          SIPP1A               SIPP1B                 Routing complete
```

```
   2          SIPP1B               SIPP1A                Routing complete

  Table Name                   : TAB2
    Class                      : Routing
    Table type                 : rtg-src-adj
    Total Call-policy Failures : 0 (0 *)
    Entry      Match Value          Destination Adjacency  Action
    -----      -----------          ---------------------  ------
    1          SIPP1A               Registrar              Routing complete
    2          SIPP1B               Registrar              Routing complete

*  Numbers in brackets refer to a call being rejected by a routing or number analysis
table because there were no matching entries in the table. This is also included in the
total figure.
```

Configuring Routing Tables

See the following sections:

- Configuring a Destination Address Table, page 7-86
- Configuring the Destination, Source Domain, and Carrier ID Tables, page 7-92
- Configuring Number Manipulation, page 7-104
- Configuring the Least Cost Table, page 7-97
- Configuring Time-Based Tables, page 7-99
- Configuring Trunk-Group ID Tables, page 7-101

Configuring a Destination Address Table

This task configures a dst-address routing table.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **call-policy-set** *policy-set-id*
5. **first-call-routing-table** *table-name*
6. **rtg-dst-address-table** *table-name*
7. **entry** *entry-id*
8. **match-address** *key* [**regex** | **string** | **digits**]
9. **prefix**
10. **dst-adjacency** *target-adjacency*
11. **action [next-table** *goto-table-name* | **complete** | **reject]**
12. **exit**
13. **entry** *entry-id*
14. **match-address** *key* [**regex** | **string** | **digits**]

15. **prefix**
16. **dst-adjacency** *target-adjacency*
17. **action [next-table** *goto-table-name* **| complete | reject]**
18. **exit**
19. **entry** *entry-id*
20. **match-address** *key* [**regex | string | digits**]
21. **prefix**
22. **dst-adjacency** *target-adjacency*
23. **action [next-table** *goto-table-name* **| complete | reject]**
24. **exit**
25. **entry** *entry-id*
26. **match-address** *key* [**regex | string | digits**]
27. **prefix**
28. **dst-adjacency** *target-adjacency*
29. **action [next-table** *goto-table-name* **| complete | reject]**
30. **exit**
31. **complete** *name*
32. **end**
33. **show**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. • Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.
Step 4	**call-policy-set** *policy-set-id* **Example:** `Router(config-sbc-sbe)# call-policy-set 1`	Enters the mode of routing policy set configuration within an SBE entity.

	Command or Action	Purpose
Step 5	**first-call-routing-table** *table-name* **Example:** `Router(config-sbc-sbe-rtgpolicy)# first-call-routing-table ROUTE-ON-DEST-NUM`	Configures the name of the first policy table to process when performing the routing stage of policy for new-call events.
Step 6	**rtg-dst-address-table** *table-name* **Example:** `Router(config-sbc-sbe-rtgpolicy)# rtg-dst-address-table MyRtgTable`	Enters the configuration mode of a routing table within the context of an SBE policy set with the entries of the table matching the dialed number (after number analysis).
Step 7	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1`	Enters the mode for configuring an entry in a routing table, creating the entry, if necessary.
Step 8	**match-address** *key* [**regex** | **string** | **digits**] **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-address 334`	Configures the match value of an entry in a routing table. To create a routing table that routes on user name, use the existing rtg-dst-address-table or rtg-src-address-table and put a textual value in the match-address field. The SBC skips number analysis and performs only routing when the SIP message contains a user name. The SBC decides that an address is a user name (as opposed to a phone number) if it contains any character other than: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, plus, hyphen, period, open-round-bracket, and close-round-bracket. When the SBC has decided that an address is a user name, the “X” in the routing tables is treated not as a wildcard character, but as a literal “X”. For example, the match value of “X” matches the username “X”, but not “A”.
Step 9	**prefix** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# prefix`	Configures the match-address of this entry to match the start of the destination address.
Step 10	**dst-adjacency** *target-adjacency* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SIP-AS540-PSTN-GW2`	Configures the destination adjacency of an entry in a routing table.

	Command or Action	Purpose
Step 11	**action [next-table** *goto-table-name* **| complete | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete`	Configures the action to take if this routing entry is chosen. Possible actions are: • Set the name of the next routing table to process if the event matches this entry. This is done using the **next-table** keyword and the *goto-table-name* argument. • Complete the action using the **complete** keyword. • Reject the indicated action using the **reject** keyword.
Step 12	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit`	Exits the **entry** mode to the **rtgtable** mode.
Step 13	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2`	Enters the mode for configuring an entry in a routing table, creating the entry, if necessary.
Step 14	**match-address** *key* **[regex | string | digits]** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-address 434`	Configures the match value of an entry in a routing table. To create a routing table that routes on user name, use the existing rtg-dst-address-table or rtg-src-address-table and put a textual value in the match-address field. The SBC skips number analysis and performs only routing when the SIP message contains a user name. The SBC decides that an address is a user name (as opposed to a phone number) if it contains any character other than: 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, plus, hyphen, period, open-round-bracket, and close-round-bracket. When the SBC has decided that an address is a user name, the “X” in the routing tables is treated not as a wildcard character, but as a literal “X”. For example, the match value of “X” matches the username “X”, but not “A”.
Step 15	**prefix** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# prefix`	Configures the match-address of this entry to match the start of the destination address.
Step 16	**dst-adjacency** *target-adjacency* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SIP-AS540-PSTN-GW1`	Configures the destination adjacency of an entry in a routing table.

	Command or Action	Purpose
Step 17	**action [next-table** *goto-table-name* **| complete | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete`	Configures the action to take if this routing entry is chosen. Possible actions are: • Set the name of the next routing table to process if the event matches this entry. This is done using the **next-table** keyword and the *goto-table-name* argument. • Complete the action using the **complete** keyword. • Reject the indicated action using the **reject** keyword.
Step 18	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit`	Exits the **entry** mode to the **rtgtable** mode.
Step 19	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 3`	Enters the mode for configuring an entry in a routing table, creating the entry, if necessary.
Step 20	**match-address** *key* **[regex | string | digits]** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-address 354`	Configures the match value of an entry in a routing table. To create a routing table that routes on user name, use the existing rtg-dst-address-table or rtg-src-address-table and put a textual value in the match-address field. The SBC skips number analysis and performs only routing when the SIP message contains a user name. The SBC decides that an address is a user name (as opposed to a phone number) if it contains any character other than: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, plus, hyphen, period, open-round-bracket, and close-round-bracket. When the SBC has decided that an address is a user name, the “X” in the routing tables is treated not as a wildcard character, but as a literal “X”. For example, the match value of “X” matches the username “X”, but not “A”.
Step 21	**prefix** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgpolicy-rtgtable -entry)# prefix`	Configures the match-address of this entry to match the start of the destination address.
Step 22	**dst-adjacency** *target-adjacency* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency H323-AS540-PSTN-GW2`	Configures the destination adjacency of an entry in a routing table.

	Command or Action	Purpose
Step 23	**action [next-table** *goto-table-name* **| complete | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete`	Configures the action to take if this routing entry is chosen. Possible actions are: • Set the name of the next routing table to process if the event matches this entry. This is done using the **next-table** keyword and the *goto-table-name* argument. • Complete the action using the **complete** keyword. • Reject the indicated action using the **reject** keyword.
Step 24	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit`	Exits the **entry** mode to the **rtgtable** mode.
Step 25	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 4`	Enters the mode for configuring an entry in a routing table, creating the entry, if necessary.
Step 26	**match-address** *key* **[regex | string | digits]** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-address 454`	Configures the match value of an entry in a routing table. To create a routing table that routes on user name, use the existing rtg-dst-address-table or rtg-src-address-table and put a textual value in the match-address field. The SBC skips number analysis and performs only routing when the SIP message contains a user name. The SBC decides that an address is a user name (as opposed to a phone number) if it contains any character other than: 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, plus, hyphen, period, open-round-bracket, and close-round-bracket. When the SBC has decided that an address is a user name, the “X” in the routing tables is treated not as a wildcard character, but as a literal “X”. For example, the match value of “X” matches the username “X”, but not “A”.
Step 27	**prefix** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# prefix`	Configures the match-address of this entry to match the start of the destination address.
Step 28	**dst-adjacency** *target-adjacency* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency H323-AS540-PSTN-GW1`	Configures the destination adjacency of an entry in a routing table.

	Command or Action	Purpose
Step 29	**action [next-table** *goto-table-name* **| complete | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete`	Configures the action to take if this routing entry is chosen. Possible actions are: • Set the name of the next routing table to process if the event matches this entry. This is done using the **next-table** keyword and the *goto-table-name* argument. • Complete the action using the **complete** keyword. • Reject the indicated action using the **reject** keyword.
Step 30	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit`	Exits the **entry** mode to the **rtgtable** mode.
Step 31	**complete** *name* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable)# complete`	Completes the full routing policy set when you have committed the full set.
Step 32	**end** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# end`	Exits rtgtable mode and enters Privileged Exec mode.
Step 33	**show** **Example:** `Router# show`	Displays the current configuration information.

Configuring the Destination, Source Domain, and Carrier ID Tables

This task configures dst-domain and src-domain and carrier ID routing tables.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **call-policy-set** *policy-set-id*
5. **rtg-src-domain-table** *table-name* | **rtg-dst-domain-table** *table-name* | **rtg-carrier-id-table** *table-name*
6. **entry** *entry-id*
7. **match-domain** *key* [**regex**] | **match-cic** *cic*
8. **edit** *action*

9. **edit-cic** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*]
10. **action [next-table** *goto-table-name* **| complete | reject]**
11. **dst-adjacency** *target-adjacency*
12. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. • Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.
Step 4	**call-policy-set** *policy-set-id* **Example:** `Router(config-sbc-sbe)# call-policy-set 1`	Enters the mode of routing policy set configuration within an SBE entity.

Step 5	**rtg-src-domain-table** *table-name* |
	rtg-dst-domain-table *table-name* |
	rtg-carrier-id-table *table-name*
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy)#`
	`rtg-src-domain-table MyRtgTable`
		Enters the configuration mode of a routing table (creating a new table if necessary) whose entries match the source or destination domains, or carrier ID respectively.
		You are not allowed to enter the submode of routing table configuration in the context of the active policy set.
		The **no** version of the command destroys the routing table. A routing table may not be destroyed if it is in the context of the active policy set.

Step 6	**entry** *entry-id*
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy-`
	`rtgtable)# entry 1`
		Enters the mode for configuring an entry in a routing table, creating the entry, if necessary.
		entry-id is a number that uniquely identifies an entry in the newly created routing table.

Step 7	**match-domain** *key* [**regex**] / **match-cic** *cic*
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy-`
	`rtgtable-entry)# match-domain ^cisco.com$`
		Creates or modifies the matching domain or carrier id code (CIC) of an entry in a routing table.
		• *key* is regular expression, not just a string.
		• *cic* is the carrier ID that matches the entry in a routing table.

	Command or Action	Purpose
Step 8	**edit** *action* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# edit del-prefix 1`	Configures a dial-string manipulation action in the routing table. You are not allowed to do this if the table is part of the active policy set. The **no** version of the command deletes the edit action of the given entry in the routing table. The **edit** command can be set to the following values: • **del-prefix** *pd*—Delete prefix *pd*, where *pd* is a positive integer specifying a number of digits to delete from the front of the dialed digit string. • **del-suffix** *sd*—Delete suffix *sd*, where *sd* is a positive integer specifying a number of digits to delete from the end of the dialed digit string. • **add-prefix** *pa*—Add prefix *pa*, where *pa* is a string of digits to add to the front of the dialed string. • **replace** *ds*—Replace *ds*, where *ds* is a string of digits that replaces the dialed string. In the example to the left, the **edit** command sets entry 1 to delete 1 digit from the beginning of the dialed string in the routing table "MyRtgTable".

Step 9	**edit-cic** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*]
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy-natable-entry)# edit-cic del-prefix 1`
		Configures a carrier identification code (CIC) manipulation action in any routing table.
		
		You are not allowed to do this if the table is part of the active policy set.
		• **del-prefix** *pd*: A positive integer specifying a number of digits to delete from the front of the carrier ID string.
		• **del-suffix** *sd*: A positive integer specifying a number of digits to delete from the end of the carrier ID string.
		• **add-prefix** *pa*: A string of digits to add to the front of the carrier ID string.
		• **replace** *ds*: A string of digits to replace the carrier ID string with.
		
		The following command sets entry 2 to delete the first digit of the carrier ID in the current routing table.
		
		If you wish to remove the carrier ID entirely from outgoing messages, you should specify a replacement string of 0000 or a prefix deletion length of 4. For example,
		
		`edit-cic del-prefix 4  OR`
		`edit-cic replace 0000`

	Command or Action	Purpose
Step 10	**action [next-table** *goto-table-name* **| complete | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete`	Configures the action to take if this routing entry is chosen. Possible actions are: • Set the name of the next routing table to process if the event matches this entry. This is done using the **next-table** keyword and the *goto-table-name* argument. • Complete the action using the **complete** keyword. • Reject the indicated action using the **reject** keyword.
Step 11	**dst-adjacency** *target-adjacency* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SIP-AS540-PSTN-GW2`	Configures the destination adjacency of an entry in a routing table.
Step 12	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit`	Exits the current mode of the configuration.

Configuring the Category Table

This task configures dst-domain and src-domain and carrier ID routing tables.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **call-policy-set** *policy-set-id*
5. **rtg-category-table** *table-name*
6. **entry** *entry-id*
7. **match-category** *word*
8. **action [next-table** *goto-table-name* **| complete | reject]**
9. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. • Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.
Step 4	**call-policy-set** *policy-set-id* **Example:** `Router(config-sbc-sbe)# call-policy-set 1`	Enters the mode of routing policy set configuration within an SBE entity.
Step 5	**rtg-category-table** *table-name* **Example:** `Router(config-sbc-sbe-rtgpolicy)# rtg-category-table MyRtgTable`	Enters the submode of configuration of a routing table whose entries match on the category within the context of an SBE policy set.
Step 6	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1`	Enters the mode for configuring an entry in a routing table, creating the entry, if necessary. *entry-id* is a number that uniquely identifies an entry in the newly created routing table.
Step 7	**match-category** *word* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-category emergency$`	Configures the match value of an entry in a routing table matching on the category.
Step 8	**action [next-table** *goto-table-name* **| complete | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action reject`	If any calls match the criterion, they are rejected.
Step 9	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit`	Exits the current mode of the configuration.

Configuring the Least Cost Table

This task configures a Least Cost routing table.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **call-policy-set** *policy-set-id*
5. **rtg-least-cost-table** *table-name*
6. **entry** *entry-id*
7. **cost** *cost*
8. **dst-adjacency**
9. **action complete**
10. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. • Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.
Step 4	**call-policy-set** *policy-set-id* **Example:** `Router(config-sbc-sbe)# call-policy-set 1`	Enters the mode of routing policy set configuration within an SBE entity.
Step 5	**rtg-least-cost-table** *table-name* **Example:** `Router(config-sbc-sbe-rtgpolicy)# rtg-least-cost-table MyRtgTable`	Enters the submode of configuration of a routing table whose entries match on the least cost within the context of an SBE policy set.

	Command or Action	Purpose
Step 6	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1`	Enters the mode for configuring an entry in a routing table, creating the entry, if necessary. *entry-id* is a number that uniquely identifies an entry in the newly created routing table.
Step 7	**cost** *cost* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# cost 50$`	Assigns a cost to the route.
Step 8	**dst-adjacency** *target-adjacency* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SIP-AS540-PSTN-GW2`	Configures the destination adjacency of an entry in a routing table.
Step 9	**action complete** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete`	Specifies that routing is complete when an entry matches this policy
Step 10	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit`	Exits the current mode of the configuration.

Configuring Time-Based Tables

This task configures dst-domain and src-domain and carrier ID routing tables.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **call-policy-set** *policy-set-id*
5. **rtg-time-table** *table-name*
6. **entry** *entry-id*
7. **match-time {[date yr** *year_low year_high* **mon** *month_low month_high* **day** *date_low date_high*] [**dow** *DoW_low DoW_high*] [**tod hr** *hour_low hour_high* **min** *minute_low minute_high*]**}**
8. **precedence** *precedence*
9. **dst-adjacency** *dst_adj*
10. **action** [**next-table** *goto-table-name* | **complete** | **reject**]
11. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. • Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.
Step 4	**call-policy-set** *policy-set-id* **Example:** `Router(config-sbc-sbe)# call-policy-set 1`	Enters the mode of routing policy set configuration within an SBE entity.
Step 5	**rtg-time-table** *table-name* **Example:** `Router(config-sbc-sbe-rtgpolicy)# rtg-time-table MyRtgTable`	Enters the submode of configuration of a routing table whose entries match on the time within the context of an SBE policy set.

	Command or Action	Purpose
Step 6	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1`	Enters the mode for configuring an entry in a routing table, creating the entry, if necessary. *entry-id* is a number that uniquely identifies an entry in the newly created routing table.
Step 7	***match-time*** *{[**date yr** year_low year_high **mon** month_low month_high **day** date_low date_high] [**dow** DoW_low DoW_high] [**tod hr** hour_low hour_high **min** minute_low minute_high]}* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time date yr 2006 2020 mon 1 12 day 1 31$`	Configures the match time of an entry. A string used to match the time and can include one or more of the following specifiers: • *date_low - date_high*—the inclusive range of dates (1-31). • **date**—date • **day**—date • *DoW_low - DoW_high*—the inclusive range of days (Sun-Mon). • **dow**—day of the week • **hr**—hour • *hour_low - hour_high*—the inclusive range of hours (0-23). • *minute_low - minute_high*—the inclusive range of minutes (0-59). • **min**—minute • **mon**—month • *month_low - month_high*—the inclusive range of months (1-12). • **tod**—time of day • **yr**—year • *year_low - year_high*—the inclusive range of years. The high values are optional and if unspecified are set equal to the low values.
Step 8	**precedence** *precedence* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 0`	Configures the precedence of the routing entry.
Step 9	**action [next-table** *goto-table-name* **| complete | reject]** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete`	Configures the action to take if this routing entry is chosen. Possible actions are: • Set the name of the next routing table to process if the event matches this entry. This is done using the **next-table** keyword and the *goto-table-name* argument. • Complete the action using the **complete** keyword. • Reject the indicated action using the **reject** keyword.

	Command or Action	Purpose
Step 10	**dst-adjacency** *dst_adj* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SIP-AS540-PSTN-GW2`	Configures the destination adjacency of an entry in a routing table.
Step 11	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit`	Exits the current mode of the configuration.

Configuring Trunk-Group ID Tables

This task configures src-trunk-group-id and dst-trunk-group-id routing tables.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **adjacency sip** *adjacency-name*
5. **tgid-routing**
6. **exit**
7. **call-policy-set** *policy-set-id*
8. **rtg-src-trunk-group-id-table** *table-name* | **rtg-dst-trunk-group-id-table** *table-name*
9. **entry** *entry-id*
10. **action** {**next-table** *goto-table-name* | **complete** | **reject**}
11. **dst-adjacency** *dst_adj*
12. **match-type** {**none** | **any** | **context** | **tgid**}
13. **tgid-context** *tgid-context-name* {tgid *tgid-name*}
14. **exit**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**configure terminal**
	
	Example:
	`Router# configure terminal`
	`Router(config)#`
		Enters global configuration mode.

Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. The *sbc-name* argument defines the name of the service.

Step 3	**sbe**
	
	Example:
	`Router(config-sbc)# sbe`
	`Router(config-sbc-sbe)#`
		Enters the mode of an SBE entity within an SBC service.

Step 4	**adjacency sip** *adjacency-name*
	
	Example:
	`Router(config-sbc-sbe)# adjacency sip adj1`
	`Router(config-sbc-sbe-adj-sip)#`
		Enters adjacency SIP configuration submode.

Step 5	**tgid-routing**
	
	Example:
	`Router(config-sbc-sbe-adj-sip)# tgid-routing`
	`Router(config-sbc-sbe-adj-sip)#`
		Enables parsing the trunk group identifier for call routing.

Step 6	**exit**
	
	Example:
	`Router(config-sbc-sbe-adj-sip)# exit`
	`Router(config-sbc-sbe)#`	

Step 7	**call-policy-set** *policy-set-id*
	
	Example:
	`Router(config-sbc-sbe)# call-policy-set 1`
	`Router(config-sbc-sbe-rtgpolicy)#`
		Enters the mode of routing policy set configuration within an SBE entity.

Step 8	**rtg-src-trunk-group-id-table** *table-name*
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy)#`
	`rtg-src-trunk-group-id-table MyRtgTable`
	`Router(config-sbc-sbe-rtgpolicy-rtgtable)#`
		Enters the submode of configuration of a routing table whose entries match on the TGID or TGID context parameters of an SBE policy set.

	Command or Action	Purpose

Step 9	**entry** *entry-id*
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1`
	`Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)#`
		Enters the mode for configuring an entry in a routing table, creating the entry, if necessary.
		
		entry-id is a number that uniquely identifies an entry in the newly created routing table.

Step 10	**action [next-table** *goto-table-name* **| complete | reject]**
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete`
	`Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)#`
		Configures the action to take if this routing entry is chosen. Possible actions are:
		• Set the name of the next routing table to process if the event matches this entry. This is done using the **next-table** keyword and the *goto-table-name* argument.
		• Complete the action using the **complete** keyword.
		• Reject the indicated action using the **reject** keyword.

Step 11	**dst-adjacency** *dst_adj*
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SIP-AS540-PSTN-GW2`
	`Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)#`
		Configures the destination adjacency of an entry in a routing table.

Step 12	**match-type** {**none | any | context | tgid**}
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-type tgid`
	`Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)#`
		Matches the entries of the routing table with the source TGID or TGID context parameter. Possible match types are:
		• **none**: Matches an entry if no TGID information is present.
		• **any**: Matches an entry if any TGID information is present.
		• **context**: Matches an entry on the TGID context.
		• **tgid**: Matches an entry on both the TGID and TGID context.

Step 13	**tgid-context** *tgid-context-name* **{tgid** *tgid-name***}**
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# tgid-context example-domain tgid trunkgroup1`
	`Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)#`
		Defines trunk-group ID context and trunk-group ID to match the entries of the routing table.

Step 14	**exit**
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit`
	`Router(config-sbc-sbe-rtgpolicy-rtgtable)#`
		Exits the current mode of the configuration.

Configuring Number Manipulation

This task enables you to specify various number manipulations that can be performed on a dialed number after a destination adjacency has been selected.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **call-policy-set** *policy-set-id*
5. **rtg-src-address-table** *table-id*
6. **rtg-src-adjacency-table** *table-id*
7. **rtg-src-account-table** *table-id*
8. **rtg-round-robin-table** *table-id*
9. **rtg-carrier-id-table** *table-id*
10. **rtg-dst-address-table** *table-id*
11. **entry** *entry-id*
12. **edit** *action*
13. **edit-cic** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*]
14. **edit-src** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*]
15. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. • Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.
Step 4	**call-policy-set** *policy-set-id* **Example:** `Router(config-sbc-sbe)# call-policy-set 1`	Enters the mode of the routing policy set configuration in the SBE mode, creating a new policy set if necessary

	Command or Action	Purpose
Step 5	**rtg-src-address-table** *table-id* **Example:** `Router(config-sbc-sbe-rtgpolicy)# rtg-src-address-table MySrcAddressTable`	Enters the configuration mode of a routing table (creating one if necessary) whose entries match the dialer's number or SIP user name within the context of an SBE policy set. You are not allowed to enter the submode of routing table configuration in the context of the active policy set. The **no** version of the command destroys the routing table. A routing table may not be destroyed if it is in the context of the active policy set.
Step 6	**rtg-src-adjacency-table** *table-id* **Example:** `Router(config-sbc-sbe-rtgpolicy)# rtg-src-adjacency-table MySrcAdjTable`	Enters the configuration mode of a routing table (creating one if necessary) within the context of an SBE policy set whose entries match the source adjacency.
Step 7	**rtg-src-account-table** *table-id* **Example:** `Router(config-sbc-sbe-rtgpolicy)# rtg-src-account-table MySrcAccTable`	Enters the configuration mode of a routing table (creating one if necessary) whose entries match the source account within the context of an SBE policy set.
Step 8	**rtg-round-robin-table** *table-id* **Example:** `Router(config-sbc-sbe-rtgpolicy)# rtg-round-robin-table MyRobinTable`	Enters the configuration mode of a policy table, whose events do not have any match-value parameters, nor next-table actions. Its actions are restricted to configuring number manipulation, as well as setting the destination adjacency. A group of adjacencies are chosen for an event if an entry in a routing table matches that event and points to a round-robin adjacency table in the next-table action.
Step 9	**rtg-carrier-id-table** *table-id* **Example:** `Router(config-sbc-sbe-rtgpolicy)# rtg-carrier-id-table MyCarrierIdTable`	Enters the configuration mode of a routing table (creating one if necessary) within the context of an SBE policy set whose entries match the carrier ID. You are not allowed to enter the mode of the routing table configuration in the context of the active policy set. The **no** version of the command destroys the routing table. A routing table may not be destroyed if it is in the context of the active policy set.
Step 10	**rtg-dst-address-table** *table-id* **Example:** `Router(config-sbc-sbe-rtgpolicy)# rtg-dst-address-table MyRtgTable`	Enters the configuration mode of a routing table (creating one if necessary) within the context of an SBE policy set whose entries match the dialed number (after number analysis) or SIP user name. You are not allowed to enter the submode of routing table configuration in the context of the active policy set. The **no** version of the command destroys the routing table. A routing table may not be destroyed if it is in the context of the active policy set.

	Command or Action	Purpose
Step 11	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1`	Enters the mode for configuring an entry in a routing table, creating the entry if necessary.
Step 12	**edit** *action* **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry) # edit del-prefix 1`	Configures a dial-string manipulation action in the routing table. You are not allowed to do this if the table is part of the active policy set. The **no** version of the command deletes the edit action of the given entry in the routing table. The **edit** command can be set to the following values: • **del-prefix** *pd*—Delete prefix *pd*, where *pd* is a positive integer specifying a number of digits to delete from the front of the dialed digit string. • **del-suffix** *sd*—Delete suffix *sd*, where *sd* is a positive integer specifying a number of digits to delete from the end of the dialed digit string. • **add-prefix** *pa*—Add prefix *pa*, where *pa* is a string of digits to add to the front of the dialed string. • **replace** *ds*—Replace *ds*, where *ds* is a string of digits that replaces the dialed string. In the example to the left, the **edit** command sets entry 1 to delete 1 digit from the beginning of the dialed string in the routing table "MyRtgTable".

Step 13	**edit-cic** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*]
	
	Example:
	`Router(config-sbc-sbe-rtgpolicy-natable-entry)# edit-cic del-prefix 1`
		Configures a CIC manipulation action in any routing table.
		You are not allowed to do this if the table is part of the active policy set.
		• **del-prefix** *pd*: A positive integer specifying a number of digits to delete from the front of the carrier ID string.
		• **del-suffix** *sd*: A positive integer specifying a number of digits to delete from the end of the carrier ID string.
		• **add-prefix** *pa*: A string of digits to add to the front of the carrier ID string.
		• **replace** *ds*: A string of digits to replace the carrier ID string with.
		The following command sets entry 2 to delete the first digit of the carrier ID in the current routing table.
		If you wish to remove the carrier ID entirely from outgoing messages, you should specify a replacement string of 0000 or a prefix deletion length of 4. For example,
		`edit-cic del-prefix 4 OR`
		`edit-cic replace 0000`

	Command or Action	Purpose
Step 14	**edit-src** [**del-prefix** *pd*] | [**del-suffix** *sd*] | [**add-prefix** *pa*] | [**replace** *ds*] **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# edit-src del-prefix 1`	Configures a source number manipulation action in the routing table. You are not allowed to do this if the table is part of the active policy set. The **no** version of the command destroys the match value. • **del-prefix** *pd*: A positive integer specifying a number of digits to delete from the front of the carrier ID string. • **del-suffix** *sd*: A positive integer specifying a number of digits to delete from the end of the carrier ID string. • **add-prefix** *pa*: A string of digits to add to the front of the carrier ID string. • **replace** *ds*: A string of digits to replace the carrier ID string with.
Step 15	**exit** **Example:** `Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit`	Exits the **entry** mode of the configuration.

Configuring Hunting

This task enables Cisco Unified Border Element (SP Edition) to hunt for other routes or destination adjacencies in case of a failure.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **adjacency sip** *adjacency-name* or **adjacency h323** *adjacency-name*
5. **hunting-trigger** *error-codes or* **hunting-trigger** *error-codes*
6. **exit**
7. **h323**
8. **hunting-mode** [**altEndps** | **multiARQ**]
9. **end**
10. **show sbc** *sbc-name* **sbe** *h323* | *sip* **hunting-trigger**
11. **show sbc** *sbc-name* **sbe** *h323* | *sip* **hunting-mode**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. • Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.

Step 4	**adjacency sip** *adjacency-name*
	or
	adjacency h323 *adjacency-name*
	
	Example:
	`Router(config-sbc-sbe)# adjacency sip test`
	
	`Router(config-sbc-sbe)# adjacency h323 test`
		Configures a destination SIP or H.323adjacency for the SBC service, and enters into adjacency sipor adjacency h323 configuration mode.
		adjacency sip—A destination SIP adjacency where the configured failure return codes cause hunting to occur. This command overrides any globally configured retry error codes.
		adjacency h323—A destination H.323 adjacency where the configured failure return codes cause hunting to occur. This command overrides any globally configured retry error codes.

Command or Action	Purpose

Step 5 **hunting-trigger** *error-codes*
or
hunting-trigger *error-codes*
 	
Example:
`Router(config-sbc-sbe-adj-sip)# hunting-trigger 415 480`
 	
(This command configures the hunting trigger for a SIP adjacency in Adjacency SIP configuration mode.)
 	
or
 	
`Router(config-sbc-sbe-adj-h323)# hunting-trigger noBandwidth`
`Router(config-sbc-sbe-adj-h323)# hunting-trigger unreachableDestination`
 	
(These commands configure the hunting trigger for an H.323 adjacency in Adjacency H.323 configuration mode.)
 	
Note If both adjacency level and SBE level hunting triggers are configured, the adjacency level takes priority.
	Configures which failure return codes cause hunting to occur, in one of the following four modes:
	• sip (global SIP scope)—use the **sip hunting-trigger** command.
	
	Note Exit (config-sbc-sbe-adj-sip) or (config-sbc-sbe-adj-h323) mode first and enter into (config-sbc-sbe) mode to configure in the global SIP scope level.
	
	• h323 (global H.323 scope)—use the **hunting-trigger** command
	• adjacency sip (destination SIP adjacency)—use the **hunting-trigger** command
	• adjacency h323 (destination H.323 adjacency)—use the **hunting-trigger** command
	
	error-codes can have the following values:
	
	In the **sip** and **adjacency sip** modes, *error-codes* represent a space-separated list of SIP numeric error codes. The examples to the left configures SIP to retry routing if it receives a “415” (media unsupported) or “480” (temporarily unavailable) error. Both error codes are set as hunting triggers. See Table 7-2 on page 7-22 for a list of SIP error codes.
	• In the **h323** and **adjacency h323** modes, *error-codes* are entered in separate commands. The following is a list of H.323 textual error codes:
	– noBandwidth—The bandwidth is taken away or the ARQ is denied.
	– unreachableDestination—The terminal cannot reach the gatekeeper for ARQ.
	– destinationRejection—The code has been rejected at destination.
	– noPermission—The callee’s gatekeeper rejects the code.
	– gatewayResources—The gateway resources are exhausted.
	– badFormatAddress—The address field in the H.225 message is not understood.
	– securityDenied—The security settings are incompatible.

	Command or Action	Purpose
		– the internally-defined value "connectFailed"—Either a releaseComplete response was received that gave no cause or any reason code for the release, or there was no response from the remote endpoint. **Note** These textual error codes apply to H.323 only. If you type **no sip hunting-trigger or no hunting-trigger**, then all error codes are cleared out. If you type **no sip hunting-trigger x y,** then just the codes **x** and **y** are removed from the configured list. **Note** In the case of the **adjacency h323** mode, enter the **noRetry** value to specify that routing should never be retried for this adjacency no matter what failure return code is received.
Step 6	**exit** **Example:** `Router(config-sbc-sbe-adj-h323)# exit`	Exits the Adjacency H.323 configuration mode and enters into SBE configuration mode.
Step 7	**h323** **Example:** `Router(config-sbc-sbe)# h323`	The **h323** command enters into the H.323 configuration mode.
Step 8	**hunting-mode** *[**altEndps**|**multiARQ**]* **Example:** `Router(config-sbc-sbe-h323)# hunting-mode multiARQ`	Configures the form of H.323 hunting to perform if H.323 hunting is triggered. • altEndps—alternateEndpoints • multiARQ—uses a nonstandard H.323 mechanism based on issuing multiple ARQs to a Gatekeeper for a single call. The **no** version of this command restores the hunting mode to the default of alternateEndpoints. It does not disable hunting completely. If the hunting mode is not defined, the default is alternateEndpoints.
Step 9	**end** **Example:** `Router(config-sbc-sbe-h323)# end`	Exits the current mode of the configuration and enters into Privileged EXEC mode.

	Command or Action	Purpose
Step 10	**show sbc** *sbc-name* **sbe** *h323|sip* **hunting-trigger** **Example:** `Router# show sbc mysbc sbe h323 hunting-trigger`	Shows the H.323 or SIP hunting triggers.
Step 11	**show sbc** *sbc-name* **sbe** ***h323|sip*** **hunting-mode** **Example:** `Router# show sbc mysbc sbe h323 hunting-mode`	Shows the H.323 hunting mode.

Activating a Routing Policy Set

This task activates a number analysis and routing policy set.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **call-policy-set default** *policy-set-id* [**priority** *priority-value*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. • Use the *sbc-name* argument to define the name of the service.

	Command or Action	Purpose
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.
Step 4	**call-policy-set default** *policy-set-id* [**priority** *priority-value*] **Example:** `Router(config-sbc-sbe)# call-policy-set default 1`	Assigns the default call-policy-set id when an administrative domain is not specified on the adjacency or the specified administrative domain does not exist. • *policy-set-id*—The call policy set number, ranging from 1 to 2147483647. The policy set must be in a complete state before it can be assigned as the default policy. • **priority**—Specifies the priority to determine which active call-policy-set to use. The SBC uses the policy set with the highest priority. • *priority*— The priority value ranging from 1 to 10 with 10 indicating highest priority. By default, priority is set to 6.

Configuring H.323 MultiARQ Hunting

This task configures Cisco Unified Border Element (SP Edition) to hunt for other H.323 routes or destination adjacencies in case of a failure.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **adjacency h323** *adjacency-name*
5. **hunting-trigger** *error-codes*
6. **hunting-mode** *mode*
7. **exit**
8. **show sbc** *sbc-name* **sbe h323 hunting-mode**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. • Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.
Step 4	**adjacency h323** *adjacency-name* **Example:** `Router(config-sbc-sbe)# adjacency h323 test`	Configures a destination H.323adjacency for the SBC service, and enters into adjacency h323 configuration mode. A destination H.323 adjacency is where the configured failure return codes cause hunting to occur. This command overrides any globally configured retry error codes.

	Command or Action	Purpose

Step 5	**hunting-trigger** *error-codes*
	
	Example:
	`Router(config-sbc-sbe-h323)# hunting-trigger noBandwidth`
	`Router(config-sbc-sbe-h323)# hunting-trigger securityDenied`
		Configures which failure return codes cause hunting to occur, in one of the following configuration modes:
		• h323 (global H.323 scope)
		• adjacency h323 (destination H.323 adjacency)
		The example to the left configures H.323 to retry routing if it receives a "noBandwidth" or "securityDenied" error codes.
		In the **h323** and **adjacency h323** configuration modes, *error-codes* are entered in separate commands. The following is a list of H.323 textual error codes:
		– noBandwidth
		– unreachableDestination
		– destinationRejection
		– noPermission
		– gatewayResources
		– badFormatAddress
		– securityDenied
		– the internally-defined value "connectFailed"
		If you type **no hunting-trigger**, all error codes are cleared out.
		Note In the case of the **adjacency h323** mode, enter the **noRetry** value to specify that routing should never be retried for this adjacency no matter what failure return code is received.

Step 6	**hunting-mode** [**altEndps**|**multiARQ**] **Example:** `Router(config-sbc-sbe-h323)# hunting-mode multiARQ`	Configures the form of hunting to perform if hunting is triggered. • altEndps—alternateEndpoints • multiARQ—uses a nonstandard H.323 mechanism based on issuing multiple ARQs to a Gatekeeper for a single call. The **no** version of this command restores the hunting mode to the default of alternateEndpoints. It does not disable hunting completely. If the hunting mode is not defined, the default is alternateEndpoints.
Step 7	**exit** **Example:** `Router(config-sbc-sbe-h323)# exit`	Exits the current mode of the configuration and enters into Privileged EXEC mode.
Step 8	**show sbc** *sbc-name* **sbe h323 hunting-mode** **Example:** `Router# show sbc mysbc sbe h323 hunting-mode`	Shows the H.323 hunting mode.

Configuring Call Admission Control Policy Sets, CAC Tables, and Global CAC Policy Sets

This optional task configures Call Admission Control policy sets, CAC tables, and assigns a global CAC policy set.

Note If you are performing this procedure to modify an active CAC policy set, see the ?$paranum>Modifying Active CAC Policy Sets? section on page 7-8 prior to performing the procedure.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **cac-policy-set averaging-period** *avg-number avg-period*
5. **cac-policy-set** *policy-set-id*
6. **first-cac-scope** *scope-name*
7. **first-cac-table** *table-name*
8. **cac-table** *table-name*
9. **table-type {policy-set | limit {***list of limit tables}}*
10. **entry** *entry-id*
11. **cac-scope** {*list of scope options}*
12. **match-value** *key*
13. **max-num-calls** *mnc*
14. **max-call-rate-per-scope** *limit* [**averaging-period** *period-num*]
15. **max-in-call-msg-rate** *limit* [**averaging-period** *period-num*]
16. **max-out-call-msg-rate** *limit* [**averaging-period** *period-num*]
17. **max-bandwidth** *mbw bwsize*
18. **callee-privacy** *callee-priv-setting*
19. **action** [next-table *goto-table-name* | cac-complete]
20. **exit**
21. **entry** *entry-id*
22. **match-value** *key*
23. **max-num-calls** *mnc*
24. **max-call-rate-per-scope** *limit* [**averaging-period** *period-num*]
25. **max-bandwidth** *mbw bwsize*
26. **transcode-deny**
27. **max-regs-rate-per-scope** *limit* [**averaging-period** *period-num*]
28. **action** [next-table *goto-table-name* | **cac-complete**]

29. **exit**
30. **exit**
31. **complete**
32. **exit**
33. **cac-policy-set global** *cac-policy-num*
34. **end**
35. **show sbc** *sbc-name* **sbe cac-policy-set** [**global**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. • Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.

Step 4	**cac-policy-set averaging-period** *avg-number avg-period*
	
	Example:
	`Router(config-sbc-sbe)# cac-policy-set averaging-period 1 100`
	`Router(config-sbc-sbe)# cac-policy-set averaging-period 2 175`
		Specifies the averaging period for rate calculations.
		• *avg-number*—The averaging period number, can be 1 or 2.
		• *avg-period*—The averaging period used by CAC in rate calculations in seconds, can range from 1 to 3600 seconds. By default, 60 seconds is configured.

Step 5	**cac-policy-set** *policy-set-id* **Example:** `Router(config-sbc-sbe)# cac-policy-set 1`	Enters the mode of CAC policy set configuration within an SBE entity, creating a new policy set if necessary. • *policy-set-id*—The call policy set number that can range from 1 to 2147483647.

	Command or Action	Purpose
Step 6	**first-cac-scope** *scope-name* **Example:** `Router(config-sbc-sbe-cacpolicy)# first-cac-scope global`	Configures the scope at which to begin defining limits when performing the admission control stage of policy. **Note** The first-cac-scope definition is only relevant if the table type configured by the first-cac-table command is a Limit table. In that case, the scope of the first-cac-table is determined by first-cac-scope. If the first-cac-table is a Policy Set table, the first-cac-scope is ignored and defaults to global. The *scope-name* argument configures the scope at which limits should be initially defined. Possible values are: • adj-group • call • category • dst-account • dst-adj-group • dst-adjacency • dst-number • global • src-account • src-adj-group • src-adjacency • src-number Features can be enabled or disabled per adjacency group through CAC configuration the same way this is done per individual adjacencies.
Step 7	**first-cac-table** *table-name* **Example:** `Router(config-sbc-sbe-cacpolicy)# first-cac-table StandardListByAccount`	Configures the name of the first policy table to process when performing the admission control stage of policy.
Step 8	**cac-table** *table-name* **Example:** `Router(config-sbc-sbe-cacpolicy)# cac-table StandardListByAccount`	Enters the mode for configuration of an admission control table (creating one if necessary) within the context of an SBE policy set.

Command or Action	Purpose
Step 9 **table-type {policy-set | limit** {*list of limit tables*}}	Configures the table type of a CAC table within the context of an SBE policy set.
Example: `Router(config-sbc-sbe-cacpolicy-cactable)# table-type policy-set`	The *list of limit tables* argument controls the syntax of the match-value fields of the entries in the table. Possible available Limit tables are: • *account*—Compare the name of the account. • *adj-group*—Compare the name of the adjacency group. • *adjacency*—Compare the name of the adjacency. • *all*—No comparison type. All events match this type. • *call-priority*—Compare with call priority. • *category*—Compare the number analysis assigned category. • *dst-account*—Compare the name of the destination account. • *dst-adj-group*—Compare the name of the destination adjacency group. • *dst-adjacency*—Compare the name of the destination adjacency. • *dst-prefix*—Compare the beginning of the dialed digit string. • *event-type*—Compare with CAC policy event types. • *src-account*—Compare the name of the source account. • *src-adj-group*—Compare the name of the source adjacency group. • *src-adjacency*—Compare the name of the source adjacency. • *src-prefix*—Compare the beginning of the calling number string. **Note** For Limit tables, the event or message or call matches only a single entry. Features can be enabled or disabled per adjacency group through CAC configuration the same way this is done per individual adjacencies. The adj-group table type matches on either source or destination adjacency group. When the policy-set keyword is specified, use the **cac-scope** command to configure the scope within each entry at which limits are applied in a CAC Policy Set table. **Note** For Policy Set tables, the event or call or message is applied to all entries in this table.

	Command or Action	Purpose
Step 10	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable)# entry 1`	Enters the mode to create or modify an entry in an admission control table.
Step 11	**cac-scope** *{list of scope options}* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# cac-scope category`	Configures the scope within each of the entries at which limits are applied in a policy set table. • *list of scope options*—Specifies one of the following strings used to match events: – **account**—Events that are from the same account. – **adjacency**—Events that are from the same adjacency. – **adj-group**—Events that are from members of the same adjacency group. – **call**—Scope limits are per single call. – **category**—Events that have same category. – **dst-account**—Events that are sent to the same account. – **dst-adj-group**—Events that are sent to the same adjacency group. – **dst-adjacency**—Events that are sent to the same adjacency. – **dst-number**—Events that have same destination. – **global**—Scope limits are global – **src-account**—Events that are from the same account. – **src-adj-group**—Events that are from the same adjacency group. – **src-adjacency**—Events that are from the same adjacency. – **src-number**—Events that have the same source number. – **sub-category**—The limits specified in this scope apply to all events sent to or received from members of the same subscriber category. – **sub-category-pfx**—The limits specified in this scope apply to all events sent to or received from members of the same subscriber category prefix. – **subscriber**—The limits specified in this scope apply to all events sent to or received from individual subscribers (a device that is registered with a Registrar server).

	Command or Action	Purpose
Step 12	**match-value** *key* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# match-value SIP-CUSTOMER-1`	Configures the match-value of an entry in a CAC Limit table. It is only relevant for Limit table types. The *key* argument is a string or a keyword based on the table type. The format of the key is determined by the Limit table type (for example, Limit event-type tables or Limit call-priority tables). For Limit event-type tables (**table-type limit** *event-type*), the match value string options are the following: • *call-update*—Compare the beginning of the calling number string. • *endpoint-reg*—Compare the name of the destination adjacency. • *new-call*—Compare the beginning of the dialed digit string. For Limit call-priority tables (**table-type limit** *call-priority*), the match value string options are the following: • *critical*—Match calls with resource priority 'critical'. • *flash*—Match calls with resource priority 'flash'. • *flash-override*—Match calls with resource priority 'flash-override'. • *immediate*—Match calls with resource priority 'immediate'. • *priority*—Match calls with resource priority 'priority'. • *routine*—Match calls with resource priority 'routine'. For all other Limit tables, enter a name or digit string • *WORD*—Name or digit string to match. (Max Size 255).
Step 13	**max-num-calls** *mnc* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-num-calls 100`	Configures the maximum number of calls of an entry in an admission control table.
Step 14	**max-call-rate-per-scope** *limit* [**averaging-period** *period-num*] **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # max-call-rate-per-scope 1000 averaging-period 2`	Configures the maximum call rate for an entry in an admission control table. • *limit*—The limit for the number of new calls per minute. The value can range from 0 to 2147483647. • **averaging-period**—Specifies the averaging-period to use in the rate calculation. By default, 1 is selected. • *period-num*—Calculates rate based on specified averaging period, ranging from 1 to 2.

	Command or Action	Purpose
Step 15	**max-in-call-msg-rate** *limit* [**averaging-period** *period-num*] **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # max-in-call-msg-rate 500 averaging-period 2`	Configures the maximum in call rate for an entry in an admission control table. • *limit*—The limit for the number of in-call messages per minute. The value can range from 0 to 2147483647. • **averaging-period**—Specifies the averaging-period to use in the rate calculation. By default, 1 is selected. • *period-num*—Calculates rate-based on specified averaging period, ranging from 1 to 2.
Step 16	**max-out-call-msg-rate** *limit* [**averaging-period** *period-num*] **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # max-out-call-msg-rate 500 averaging-period 2`	Configures the maximum out call rate for an entry in an admission control table. • *limit*—The limit for the number of new calls per minute. The value can range from 0 to 2147483647. • **averaging-period**—Specifies the averaging-period to use in the rate calculation. By default, 1 is selected. • *period-num*—Calculates rate-based on specified averaging period, ranging from 1 to 2.
Step 17	**max-bandwidth** *mbw bwsize* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # max-bandwidth 1000000 bps`	Configures the maximum bidirectional bandwidth for an entry in an admission control table. For example, if a max-bandwidth value is configured, the SBC allows half of this value in each direction. The *mbw* argument is a positive integer specifying the total maximum rate at which call media should be admitted in both directions (in bytes per second). The *bwsize argument specifies the transfer size to which mbw refers. Possible values are:* • bps • Kbps • Mbps • Gbps
Step 18	**callee-privacy [***callee-priv-setting***]** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # callee-privacy never`	Configures the level of privacy processing to perform on messages sent from callee to caller. The *callee_priv_setting* argument indicates the specific callee privacy setting. Possible values are: • never—Indicates to never hide identity. • account-boundary—Indicates to hide identity only if caller is different account from callee. • always—Indicates to always hide identity.

	Command or Action	Purpose
Step 19	**action [next-table** *goto-table-name* | **cac-complete]** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # action cac-complete`	Configures the action to perform after this entry in an admission control table. Possible actions are: • Identify the next CAC table to process using the **next-table** keyword and the *goto-table-name argument.* • Stop processing for this scope using the **cac-complete** keyword.
Step 20	**exit** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # exit`	Exits from **entry** to **cactable** mode.
Step 21	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable)# entry 2`	Enters the mode to create or modify an entry in an admission control table.
Step 22	**match-value** *key* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # match-value SIP-CUSTOMER-2`	Configures the match-value of an entry in a CAC Limit table. The *key* argument is a string used to match events. The format of the key is determined by the Limit table type (for example, Limit event-type tables or Limit call-priority tables). See the **match-value** command page for more details.
Step 23	**max-num-calls** *mnc* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # max-num-calls 110`	Configures the maximum number of calls of an entry in an admission control table.
Step 24	**max-call-rate-per-scope** *limit* [**averaging-period** *period-num*] **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # max-call-rate-per-scope 1000 averaging-period 2`	Configures the maximum call rate for an entry in an admission control table. • *limit*—The limit for the number of new calls per minute. The value can range from 0 to 2147483647. • **averaging-period**—Specifies the averaging-period to use in the rate calculation. By default, 1 is selected. • *period-num*—Calculates rate-based on specified averaging period, ranging from 1 to 2.

	Command or Action	Purpose
Step 25	**max-bandwidth** *mbw bwsize* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-bandwidth 2000000 bps`	Configures the maximum bidirectional bandwidth for an entry in an admission control table. For example, if a max-bandwidth value is configured, the SBC allows half of this value in each direction. The *mbw* argument is a positive integer specifying the total maximum rate at which call media should be admitted in both directions (in bytes per second). The *bwsize argument specifies the transfer size to which mbw refers. Possible values are:* • bps • Kbps • Mbps • Gbps
Step 26	**transcode-deny** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# transcode-deny`	Forbids transcoding for this entry in an admission control table.
Step 27	**max-regs-rate-per-scope** *limit* [**averaging-period** *period-num*] **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-regs-rate-per-scope 300 averaging-period 2`	Configures the maximum call number of subscriber registrations for an entry in an admission control table. • *limit*—The limit for the number of new calls per minute. The value can range from 0 to 2147483647. • **averaging-period**—Specifies the averaging-period to use in the rate calculation. By default, 1 is selected. • *period-num*—Calculates rate-based on specified averaging period, ranging from 1 to 2.
Step 28	**action** [**next-table** *goto-table-name* | **cac-complete]** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete`	Configures the action to perform after this entry in an admission control table. Possible actions are: • Identify the next CAC table to process using the **next-table** keyword and the *goto-table-name argument.* • Stop processing for this scope using the **cac-complete** keyword.
Step 29	**exit** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# exit`	Exits from **entry** to **cactable** mode.
Step 30	**exit** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable)# exit`	Exits from **cactable** to **cacpolicy** mode.

	Command or Action	Purpose
Step 31	**complete** **Example:** `Router(config-sbc-sbe-cacpolicy)# complete`	Completes the CAC policy set when you have committed the full set.
Step 32	**exit** **Example:** `Router(config-sbc-sbe-cacpolicy)# exit`	Exits the SBE CAC policy mode.
Step 33	**cac-policy-set global** *policy-num* **Example:** `Router(config-sbc-sbe)# cac-policy-set global 23`	Activates the global CAC policy set. The CAC policy set must be in a complete state before it can be assigned as the default policy. • *policy-num*—The call policy set number, ranging from 1 to 2147483647. The policy set must be in a complete state before it can be assigned as the default policy.
Step 34	**end** **Example:** `Router(config-sbc-sbe)# end`	Exits the SBE mode to Privileged EXEC mode.
Step 35	**show sbc** *sbc-name* **sbe cac-policy-set** [**global**] **Example:** `Router# show sbc mySBC sbe cac-policy-set`	Displays details of the CAC policy sets configured on the SBC. • *sbc-name*—Defines the name of the SBC service. • **global**—Lists the information pertaining to the global CAC policy set.

The following example shows the output of the **show sbc sbe cac-policy-set** command:

```
Router# show sbc mySBC sbe cac-policy-set
SBC Service "mySBC"
CAC Averaging period 1: 100 sec
CAC Averaging period 2: 1500 sec

CAC Policy Set 2
  Global policy set: Yes
  First CAC table: 1
  First CAC scope: src-adjacency

  Table name: 1
    Table type: limit adjacency
    Total call setup failures (due to non-media limits): 0
    Entry   Match value                        Action                 Failures
    -----   -----------                        ------                 --------
    1       SIPP1A                             Complete                      0
    2       SIPP1B                             Complete                      0

CAC Policy Set 12
  Global policy set: No
  First CAC table: 1
  First CAC scope: global

  Table name: 1
    Table type: limit adjacency
    Total call setup failures (due to non-media limits): 0
```

```
    Entry   Match value                          Action                 Failures
    -----   -----------                          ------                 --------
    2       SIPP1B                               Complete                      0

CAC Policy Set 21
  Global policy set: No
  First CAC table: 1
  First CAC scope: src-adjacency

  Table name: 1
    Table type: limit adjacency
    Total call setup failures (due to non-media limits): 0

CAC Policy Set 22
  Global policy set: No
  First CAC table:
  First CAC scope: global

  Table name: table1
    Table type: limit adjacency
    Total call setup failures (due to non-media limits): 0
    Entry   Match value                          Action                 Failures
    -----   -----------                          ------                 --------
    1                                            Complete                      0

CAC Policy Set 25
  Global policy set: No
  First CAC table: TBL2
  First CAC scope: global

  Table name: Table2
    Table type: limit adjacency
    Total call setup failures (due to non-media limits): 0
    Entry   Match value                          Action                 Failures
    -----   -----------                          ------                 --------
    1       SIPP                                 Complete                      0
```

The following example shows the output of the **show sbc sbe cac-policy-set global** command:

```
Router# show sbc mySBC sbe cac-policy-set global
SBC Service "mySBC"
CAC Averaging period 1: 100 sec
CAC Averaging period 2: 1500 sec

CAC Policy Set 2
  Global policy set: Yes
  First CAC table: 1
  First CAC scope: src-adjacency

  Table name: 1
    Table type: limit adjacency
    Total call setup failures (due to non-media limits): 0
    Entry   Match value                          Action                 Failures
    -----   -----------                          ------                 --------
    1       SIPP1A                               Complete                      0
    2       SIPP1B                               Complete                      0
```

Configuring Privacy Service

This section describes the tasks to configure the privacy service on a CAC policy set, adjacencies, and number analysis table:

Configuring Privacy Service on a CAC Policy Set

This task shows how to configure the privacy service on a CAC policy set.

Note The **caller** and **callee** commands have been used in this procedure. In some scenarios, the **branch** command can be used as an alternative to the **caller** and **callee** command pair. The **branch** command has been introduced in Release 3.5.0. See the ?$paranum>Configuring Directed Nonlimiting CAC Policies? section on page 7-37 for information about this command.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **cac-policy-set policy-set-id**
5. **cac-table table-name**
6. **table-type {policy-set | limit {list of limit tables}}**
7. **entry entry-id**
8. **caller-privacy edit-privacy-request {pass | strip | insert | replace | sip {strip {all | critical | header | id | none | session | token word | user} | insert {critical | header | id | none | session | token** *word* **| user}}}**
9. **callee-privacy edit-privacy-request {pass | strip | insert | replace | sip {strip {all | critical | header | id | none | session | token word | user} | insert {critical | header | id | none | session | token** *word* **| user}}}**
10. **caller-privacy privacy-service {adj-trust-boundary | always | never}**
11. **callee-privacy privacy-service {adj-trust-boundary | always | never}**
12. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the SBC service mode. Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the SBE entity mode within an SBC service.
Step 4	**cac-policy-set** *policy-set-id* **Example:** `Router(config-sbc-sbe)# cac-policy-set 1`	Enters the CAC policy set configuration mode within an SBE entity, creating a new policy set, if necessary. • *policy-set-id*—The call policy set number that can range from 1 to 2147483647.
Step 5	**cac-table** *table-name* **Example:** `Router(config-sbc-sbe-cacpolicy)# cac-table StandardListByAccount`	Enters the admission control table configuration mode (creating one, if necessary) within the context of an SBE policy set.

Command or Action	Purpose
Step 6 **table-type {policy-set | limit** {*list of limit tables*}} **Example:** `Router(config-sbc-sbe-cacpolicy-cactable)# table-type policy-set`	Configures the table type of a CAC table within the context of an SBE policy set. The *list of limit tables* argument controls the syntax of the match-value fields of the entries in the table. *list of limit tables* values are: • *account*—Compares the name of the account. • *adj-group*—Compares the name of the adjacency group. • *adjacency*—Compares the name of the adjacency. • *all*—No comparison type. All the events match this type. • *call-priority*—Compares with call priority. • *category*—Compares the number analysis-assigned category. • *dst-account*—Compares the name of the destination account. • *dst-adj-group*—Compares the name of the destination adjacency group. • *dst-adjacency*—Compares the name of the destination adjacency. • *dst-prefix*—Compares the beginning of the dialed digit string. • *event-type*—Compares with CAC policy event types. • *src-account*—Compares the name of the source account. • *src-adj-group*—Compares the name of the source adjacency group. • *src-adjacency*—Compares the name of the source adjacency. • *src-prefix*—Compares the beginning of the calling number string. **Note** For Limit tables, the event, message, or call matches only a single entry. Features can be enabled or disabled per adjacency group through CAC configuration the same way this is done per individual adjacency. The adj-group table type matches on either the source adjacency group or the destination adjacency group. After the **policy-set** keyword is specified, use the **cac-scope** command to configure the scope within each entry in which limits are applied in a CAC policy set table. **Note** In Policy Set tables, the event, call, or message is applied to all the entries.

	Command or Action	Purpose
Step 7	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable)# entry 1`	Enters the CAC table entry mode to create or modify an entry in an admission control table.

Command or Action	Purpose
Step 8 **caller-privacy edit-privacy-request** {**pass** | **strip** | **insert** | **replace** | **sip** {**strip** {**all** | **critical** | **header** | **id** | **none** | **session** | **token** *word* | **user**} | **insert** {**critical** | **header** | **id** | **none** | **session** | **token** *word* | **user**}}} **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # caller-privacy edit-privacy-request strip`	Edits and updates the privacy indications provided by the user: • **insert**—Inserts privacy restrictions: – **SIP**—Inserts Privacy:header;session;user;id;critical if the header is not present already. – **H323**—Sets presentation indicator from allowed to restricted. • **pass**—Passes on the privacy header or the presentation indicators. • **replace**—Replaces privacy restrictions: – **SIP**—Replaces Privacy:header;session;user;id;critical, except when none has been requested. – **H323**—Sets the presentation indicator to restricted. • **strip**—Removes all the privacy restrictions: – **SIP**—Removes the Privacy header. – **H323**—Sets the presentation indicator to allowed. • **sip**—Specifies the following SIP settings. This allows greater control and overrides all generic actions: – **insert**—Inserts privacy tokens into the Privacy header. – **strip**—Removes privacy tokens from the Privacy header. • **critical**—Specifies the call to be discontinued if privacy cannot be achieved in the SIP Privacy header. • **header**—Obscures all the header information, which is related to the user, from the SIP Privacy header. • **id**—Removes ID headers from the SIP Privacy header. • **none**—Privacy is not applied to the call. • **session**—Specifies media privacy for the session in the SIP Privacy header. No media bypass is performed. • **token**—Specifies the nonstandard user-defined privacy token in the SIP Privacy header. • *word*—Specifies the user-defined privacy token. • **user**—Removes all nonessential header information, which is related to the user, from the SIP Privacy header. By default, the privacy setting value is set to **pass**.

Command or Action	Purpose
Step 9 **callee-privacy edit-privacy-request** {**pass** | **strip** | **insert** | **replace** | **sip** {**strip** {**all** | **critical** | **header** | **id** | **none** | **session** | **token** *word* | **user**} | **insert** {**critical** | **header** | **id** | **none** | **session** | **token** *word* | **user**}}} **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# callee-privacy edit-privacy-request strip`	Edits and updates privacy indications provided by the user: • **insert**—Inserts privacy restrictions: – **SIP**—Inserts Privacy:header;session;user;id;critical if the header is not present already. – **H323**—Sets presentation indicator from allowed to restricted. • **pass**—Passes on the privacy header or the presentation indicators. • **replace**—Replaces privacy restrictions: – **SIP**—Replaces Privacy:header;session;user;id;critical, except when none has been requested. – **H323**—Sets the presentation indicator to restricted. • **strip**—Removes all the privacy restrictions: – **SIP**—Removes the Privacy header. – **H323**—Sets the presentation indicator to allowed. • **sip**—Specifies the following SIP settings. This allows greater control and overrides all generic actions: – **insert**—Inserts privacy tokens into the Privacy header. – **strip**—Removes privacy tokens from the Privacy header. • **critical**—Specifies the call to be discontinued if privacy cannot be achieved in the SIP Privacy header. • **header**—Obscures all the header information, which is related to the user, from the SIP Privacy header. • **id**—Removes ID headers from the SIP Privacy header. • **none**—Privacy is not applied to the call. • **session**—Specifies media privacy for the session in the SIP Privacy header. No media bypass is performed. • **token**—Specifies the nonstandard user-defined privacy token in the SIP Privacy header. • *word*—Specifies the user-defined privacy token. • **user**—Removes all nonessential header information, which is related to the user, from the SIP Privacy header. By default, the privacy setting value is set to **pass**.

	Command or Action	Purpose
Step 10	**caller-privacy privacy-service** {**adj-trust-boundary** | **always** | **never**} **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # caller-privacy privacy-service always`	Configures privacy settings according to RFC3323, RFC3325, and/or setting of the H.323 presentation restriction settings in a given entry in the admission control table: • **adj-trust-boundary**—Specifies the adjacency privacy trust level to determine if the privacy service is required. • **always**—Provides privacy service always, if requested by the user. • **never**—Never provides privacy service even if requested by the user. By default, the privacy setting value is set to **adj-trust-boundary**.
Step 11	**callee-privacy privacy-service** {**adj-trust-boundary** | **always** | **never**} **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # callee-privacy privacy-service adj-trust-boundary`	Configures privacy settings according to RFC3323, RFC3325, and/or setting of H.323 presentation restriction settings in a given entry in the admission control table: • **adj-trust-boundary**—Specifies the adjacency privacy trust level to determine if the privacy service is required. • **always**—Provides privacy service always, if requested by the user. • **never**—Never provides privacy service even if requested by the user. By default, the privacy setting value is set to **adj-trust-boundary**.
Step 12	**end** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # end`	Exits from the CAC table entry configuration mode and enters the Privileged EXEC mode.

Configuring Privacy Service on Adjacencies

This task shows how to configure the privacy service on the SIP and H323 adjacencies.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **adjacency sip adjacency-name**
5. **privacy [inherit-profile | trusted | untrusted]**
6. **exit**
7. **adjacency h323 adjacency-name**

8. **allow private info**
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the SBC service mode. Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the SBE entity mode within an SBC service.
Step 4	**adjacency sip** *adjacency-name* **Example:** `Router(config-sbc-sbe)# adjacency sip SIPP`	Enters the SBE SIP adjacency mode. Use the *adjacency-name* argument to define the name of the service.
Step 5	**privacy** [**inherit-profile** | **trusted** | **untrusted**] **Example:** `Router(config-sbe-adj-sip)# privacy trusted`	Configures the trust level for determining whether the privacy service should be applied: • **inherit-profile**—Specifies that the trust level for determining whether privacy services are required is derived from the adjacencies inherit-profile. • **trusted**—Specifies that the adjacency is trusted and privacy services do not have to be applied. • **untrusted**—Specifies that the adjacency is not trusted and requires privacy services to be applied. By default, the trust level is set to **inherit-profile**.
Step 6	**exit** **Example:** `Router(config-sbe-adj-sip)# exit`	Exits the SIP adjacency mode and enters the SBE mode.
Step 7	**adjacency h323** *adjacency-name* **Example:** `Router(config-sbc-sbe)# adjacency h323 test`	Configures a destination H.323 adjacency for the SBC service, and enters into H. 323 adjacency configuration mode. A destination H.323 adjacency is where the configured failure return codes cause hunting to occur. This command overrides any globally configured retry error codes.

	Command or Action	Purpose
Step 8	**allow private info** **Example:** `Router(config-sbe-adj-h323)# allow private info`	Configures the H.323 adjacency to allow private information to be sent. By default, the H.323 adjacency does not send the private information of a user.
Step 9	**end** **Example:** `Router(config-sbe-adj-h323)# end`	Exits from a H.323 adjacency configuration mode and entry the Privileged EXEC mode.

Configuring a Number Analysis Table

This task shows how to configure a number analysis table to detect anonymity.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **call-policy-set** *policy-set-id*
5. **na-src-name-anonymous-table** *table-name*
6. **entry** *entry-id*
7. **match-anonymous** [**false** | **true**]
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the SBC service mode. Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the SBE entity mode within an SBC service.
Step 4	**call-policy-set** *policy-set-id* **Example:** `Router(config-sbc-sbe)# call-policy-set 1`	Enters the routing policy set configuration mode within an SBE entity.
Step 5	**na-src-name-anonymous-table** *table-name* **Example:** `Router(config-sbc-sbe-rtgpolicy)# na-src-name-anonymous-table NameTable`	Enters the configuration mode of a number analysis table to determine whether the display name or presentation number is anonymous.
Step 6	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-rtgpolicy-natable)# entry 1`	Enters the number analysis table entry mode for configuring an entry in a number analysis table, creating the entry, if necessary.
Step 7	**match-anonymous** [**false** | **true**] **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# match-anonymous false`	Matches the display name or presentation number to Anonymous in the na-src-name-anonymous-table number analysis table. • false—Specifies the display name or presentation number as not anonymous. • true—Specifies the display name or presentation number as anonymous.
Step 8	**end** **Example:** `Router(config-sbc-sbe-rtgpolicy-natable-entry)# end`	Exits the number analysis table entry mode and enters the Privileged EXEC mode.

Configuring Multiple SBC Media Bypass

This task shows how to configure the Multiple SBC Media Bypass feature. The steps to configure the renegotiation of media bypass after a session refreshes are also included in this task.

Note The **caller** and **callee** commands have been used in this procedure. In some scenarios, the **branch** command can be used as an alternative to the **caller** and **callee** command pair. The **branch** command has been introduced in Release 3.5.0. See the ?$paranum>Configuring Directed Nonlimiting CAC Policies? section on page 7-37 for information about this command.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **adjacency sip** *adjacency-name*
5. **media bypass** {**max-data-len** *data-length* | **tag** *sequence-number tag-name*}
6. **exit**
7. **cac-policy-set policy-set-id**
8. **cac-table table-name**
9. **table-type {policy-set | limit {list of limit tables}}**
10. **entry entry-id**
11. **match-value** *key*
12. **media bypass type** [**all** | **none** | **full** [**hairpin partial**] | **hairpin** [**full partial**] | **partial** [**full hairpin**]
13. **caller media bypass** {**enable** | **disable**}
14. **callee media bypass** {**enable** | **disable**}
15. **action** [**next-table** *goto-table-name* | **cac-complete**]
16. **exit**
17. **entry** *entry-id*
18. **session-refresh renegotiation {allow | suppress}**
19. **end**
20. **show sbc** *sbc-name* **sbe cac-policy-set** *id* **table** *name* **entry** *entry*
21. **show sbc** *sbc-name* **sbe adjacencies** *adjacency-name* **detail**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the SBC service mode. Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the SBE entity mode within an SBC service.
Step 4	**adjacency sip** *adjacency-name* **Example:** `Router(config-sbc-sbe)# adjacency sip access`	Enters the SBE SIP adjacency mode. • Use the adjacency-name argument to define the name of the service.
Step 5	**media bypass** {**max-data-len** *data-length* | **tag** *sequence-number tag-name*} **Example:** `Router(config-sbc-sbe-adj-sip)# media bypass tag 1 TAG1`	Configures the multiple SBC media bypass feature on a SIP adjacency: • **max-data-len**—Specifies the maximum length of the multiple SBC media bypass data that can be transmitted on outbound signaling messages on an adjacency. • **tag**—Specifies the tag that can be used to control groups to which endpoints on the adjacency belong to the multiple SBC media bypass feature. • *data-length*—Specifies the maximum multiple SBC media bypass data length in bytes that can range from 100 to 2048. By default, *data-length* is set to 1000 bytes. • *sequence-number*—Specifies the sequence number for a media bypass tag in the tag list. The tag list is formed from the set of tags ordered according to their sequence number. The sequence number can range from 1 to 20. • *tag-name*—Specifies the name of the multiple SBC media bypass tag. The total length of all tags in an adjacency cannot exceed 255 characters. Each tag must consist of alphabets, numerals, and special characters. All printable characters other than comma, semi-colon & space. **Note** Media bypass is not supported for H.323 calls.

	Command or Action	Purpose
Step 6	**exit** **Example:** `Router(config-sbc-sbe-adj-sip)# exit`	Exits the adjacency SIP mode and enters the SBE entity mode.
Step 7	**cac-policy-set** *policy-set-id* **Example:** `Router(config-sbc-sbe)# cac-policy-set 1`	Enters the CAC policy set configuration mode within an SBE entity, creating a new policy set if necessary. • *policy-set-id*—The call policy set number that can range from 1 to 2147483647.
Step 8	**cac-table** *table-name* **Example:** `Router(config-sbc-sbe-cacpolicy)# cac-table MyTable`	Enters the admission control table configuration mode (creating one if necessary) within the context of an SBE policy set.
Step 9	**table-type {policy-set | limit** {*list of limit tables*}} **Example:** `Router(config-sbc-sbe-cacpolicy-cactable)# table-type src-adjacency`	Configures the limit of the table types to be matched by the match-value command. For the multiple SBC media bypass feature, use the following table type: • *src-adjacency*—Compare the name of the source adjacency.
Step 10	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable)# entry 1`	Enters the mode to create or modify an entry in an admission control table.
Step 11	**match-value** *key* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# match-value access`	Configures the match-value of an entry in a CAC Limit Table. • *key*—Specifies the keyword used to match events. The format of the key is determined by the table-type limit.

	Command or Action	Purpose
Step 12	**media bypass type** [**all** | **none** | **full** [**hairpin partial**] | **hairpin** [**full partial**] | **partial** [**full hairpin**] **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # media bypass type full hairpin`	Configures the multiple SBC media bypass feature for CAC policy set. • **all**—Enables all, such as partial, hairpin, and full types of media bypass for the CAC table entry. • **none**—Disables all types of media bypass for the CAC table entry. • **full**—Enables media bypass on the SBC if adjacent and non-adjacent downstream and upstream hops have direct media connectivity, common tags in bypass tag list or with same VPN. • **hairpin**—Enables media bypass for the hairpin calls. • **partial**—Enables media bypass if the SBC is a member of a group of SBCs that share the same IP realm and if even one SBC within that group is on the media path. **Note** If the media bypass type is explicitly configured to be partial, only IP realm and VPN configuration on the adjacency can be used to determine whether media bypass is possible. Because media bypass tags are not used, the VPN names must be globally unique across all the SBCs for partial media bypass to work.
Step 13	**caller media bypass** {**enable** | **disable**} **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # caller media bypass enable`	Enables or disables the multiple SBC media bypass feature on the caller side. • **enable**—Enables the multiple SBC media bypass feature on the caller side. • **disable**—Disables the multiple SBC media bypass feature on the caller side.
Step 14	**callee media bypass** {**enable** | **disable**} **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # callee media bypass enable`	Enables or disables the multiple SBC media bypass feature on the callee side. • **enable**—Enables the multiple SBC media bypass feature on the callee side. • **disable**—Disables the multiple SBC media bypass feature on the callee side.
Step 15	**action** [**next-table** *goto-table-name* | **cac-complete**] **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry) # action cac-complete`	Configures the action to be performed after this entry, in an admission control table. Possible actions are: • Identify the CAC table to be processed next using the **next-table** keyword and the *goto-table-name* argument. • Stop the processing action for this scope using the **cac-complete** keyword.

	Command or Action	Purpose
Step 16	**exit** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete`	Configures the action to be performed after this entry, in an admission control table. Possible actions are: • Identify the CAC table to be processed next using the **next-table** keyword and the *goto-table-name* argument. • Stop the processing action for this scope using the **cac-complete** keyword.
Step 17	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable)# entry 2`	Enters the mode to create or modify an entry in an admission control table.
Step 18	**session-refresh renegotiation {allow | suppress}** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# session-refresh renegotiation suppress`	Depending on the option that you select, one of the following actions is configured: • **allow**—Specifies that an offer that contains duplicate SDP must be processed using the normal offer-answer rules. Media reservations can change, and interworking functions can be renegotiated. • **suppress**—Specifies that an offer that contains duplicate SDP must be processed using the session refresh variant of the offer-answer rules. Media reservations are not changed, and interworking functions are not renegotiated. The SBC forwards the last sent offer or answer regardless of the offer or answer that was received. The default is that the session refresh strategy for the call is not affected by this CAC policy entry.
Step 19	**end** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# end`	Exits from the CAC table entry configuration mode and enters the Privileged EXEC mode.
Step 20	**show sbc** *sbc-name* **sbe cac-policy-set** *id* **table** *name* **entry** *entry* **Example:** `Router# show sbc mysbc sbe cac-policy-set 1 table MyTable entry 1`	Displays detailed information about a specific entry in a CAC policy table.
Step 21	**show sbc** *sbc-name* **sbe adjacencies** *adjacency-name* **detail** **Example:** `Router# show sbc sbe mySBC sbe adjacencies access detail`	Displays all the detailed field outputs for the specified SIP adjacency.

Configuring Common IP Address Media Bypass

This procedure shows how to configure the Common IP Address Media Bypass feature.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **adjacency sip** *adjacency-name*
5. **media bypass auto-nat-tag-gen**
6. **end**
7. **show sbc** *sbc-name* **sbe adjacencies** *adjacency-name* **detail**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.
Step 4	**adjacency sip** *adjacency-name* ***Example:*** `Router(config-sbc-sbe)# adjacency sip access-side-1`	Enters the mode of an SBE SIP adjacency. • *adjacency-name*—Name of the adjacency.
Step 5	**media bypass auto-nat-tag-gen** **Example:** `Router(config-sbc-sbe-adj-sip)# media bypass auto-nat-tag-gen`	Configures the Common IP Address Media Bypass feature to generate a media bypass tag for the registered endpoints that are behind a NAT device associated with this adjacency.

	Command or Action	Purpose
Step 6	**end** **Example:** `Router(config-sbc-sbe-adj-sip)# end`	Exits the SBE SIP adjacency mode, and enters the privileged EXEC mode.
Step 7	**show sbc** *sbc-name* **sbe adjacencies** *adj-name* **detail** **Example:** `Router# show sbc mySBC sbe adjacencies access-side-1 detail`	Shows the configuration details of the specified adjacency.

Activating a CAC Policy Set

This task activates a global CAC policy set.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **cac-policy-set global** *policy-set-id*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the mode of an SBC service. Use the *sbc-name* argument to define the name of the service.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the mode of an SBE entity within an SBC service.
Step 4	**cac-policy-set global** *policy-set-id* **Example:** `Router(config-sbc-sbe)# cac-policy-set global 1`	Activates the global CAC policy set within an SBE entity. • *policy-set-id*—The call policy set number that can range from 1 to 2147483647.

Configuring Asymmetric Payload Types

This task configures SBC to allow asymmetric payload types.

SUMMARY STEPS

1. **configure** *terminal*
2. **sbc** *sbc-name*
3. **sbe**
4. **cac-policy-set** *policy-set-id*
5. **first-cac-table** *table-name*
6. **cac-table** *table-name*
7. **table-type policy-set**
8. **entry** *entry-id*
9. **action cac-complete**
10. **payload-type asymmetric allowed**
11. **complete**
12. **cac-policy-set global** *policy-set-id*
13. **end**
14. **show sbc sbc-name sbe cac-policy-set**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `Router# configure`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mySbc`	Enables entry into the mode of an SBC service. Use the *sbc-name* argument to define the name of an SBC.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enables entry into the mode of an SBE entity within an SBC service.
Step 4	**cac-policy-set** *policy-set-id* **Example:** `Router(config-sbc-sbe)# cac-policy-set 1`	Enables entry into the mode of the CAC policy.

	Command or Action	Purpose
Step 5	**first-cac-table** *table-name* **Example:** `Router(config-sbc-sbe-cacpolicy)# first-cac-table first_policy_table`	Configures the name of the first policy table to be processed when performing the admission control stage of the CAC policy.
Step 6	**cac-table** *table-name* **Example:** `Router(config-sbc-sbe-cacpolicy)# cac-table first_policy_table`	Enables entry into the mode for configuring an admission control table (or creating one, if necessary) within the context of an SBE policy set.
Step 7	**table-type {policy-set | limit {***list of limit tables***}}** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable)# table-type policy-set`	Configures the table type of a CAC Policy table within the context of an SBE policy set.
Step 8	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable)# entry 1`	Enables entry into the mode to create or modify an entry in an admission control table.
Step 9	**action [next-table** *goto-table-name* **| cac-complete]** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete`	Configures the action to be performed after this entry, in an admission control table. Possible actions are: • Identify the next CAC table to be processed using the **next-table** keyword and the *goto-table-name* argument. • Stop the processing for this scope using the **cac-complete** keyword.
Step 10	**payload-type asymmetric allowed** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# payload-type asymmetric allowed`	Configures SBC to allow asymmetric payload types.
Step 11	**complete** **Example:** `Router(config-sbc-sbe-cacpolicy)# complete`	Completes the CAC policy.
Step 12	**cac-policy-set global** *policy-set-id* **Example:** `Router (config-sbc-sbe)# cac-policy-set global 1`	Activates the global CAC policy set within an SBE entity.

	Command or Action	Purpose
Step 13	**end** **Example:** `Router(config-sbc-sbe-cacpolicy)# end`	Enables exit from the CAC policy set configuration mode and entry into the Privileged EXEC mode.
Step 14	**show sbc** *sbc-name* **sbe cac-policy-set** *id* **table** *name* **entry** *entry* **Example:** `Router# show sbc mysbc sbe cac-policy-set 1 table standard_policy_list entry 1`	Displays detailed information for a specific entry in a CAC policy table, including any restricted codecs.

Limiting Resource Usage

New router features, such as transcoding, transrating, and inband DTMF interworking, have been introduced in earlier releases. If no limits are set on the number of calls that use the resources provided by these features, overload conditions may occur and the router may stop responding. You can configure limits on resource usage to prevent the occurrence of overload conditions. This is one of the areas in which Cisco Unified Border Element (SP Edition) policies can be applied.

Note The Limiting Resource Usage feature has been introduced in Release 3.4S.

You can configure media policies to specify maximum levels of usage for the following:

- Number of audio streams using transcoding
- Number of audio streams using transrating
- Number of video streams using transcoding
- Number of audio streams using inband DTMF interworking
- Number of streams using SRTP encryption and decryption
- Number of registered subscribers using IPSec encryption and decryption on the signaling link to the SBC
- Number of calls made by subscribers who are using IPSec-protected signaling
- Total number of video and audio streams using transcoding, transrating, inband DTMF interworking, and SRTP encryption and decryption—weighted by the costs assigned to each of these resources.

  Table 7-10 lists the default resource costs. You can modify these default resource costs.

Table 7-10 Default Resource Costs

Resource	Default Resource Cost
Audio transcoding	10
Audio transrating	6
Video transcoding	50

Table 7-10 Default Resource Costs

Resource	Default Resource Cost
Inband DTMF interworking	4
SRTP encryption and decryption	15

At run time, the total resource usage value is calculated for each incoming call using the resource costs configured for the resources requested by the call. This calculated value is then compared with the maximum total resource usage value that you have configured. If the calculated value is more than the configured value, the media policy rejects the call. This means that the call either fails or is directed to a different message gateway or signaling route.

After you define a media policy, you can apply it in one of the following ways:

- As a CAC policy

  For example, call-scoped policies restrict resource usage for a particular call. In contrast, adjacency-scoped policies restrict resource usage at the adjacency level.

- As a media gateway policy

  Media policies applied at the media gateway level restrict resource usage for the media gateway.

After you apply a media policy, you can view the resource usage of each resource for which you have specified a limit in the media policy. For example, you can view the number of media streams that are being video-transcoded by the message gateway on which you have applied the media policy.

The following sections describe the procedures for limiting resource usage:

- Configuring Resource Costs for Transcoding, Transrating, Inband DTMF Interworking, and SRTP Encryption and Decryption, page 7-146
- Configuring Usage Limits for Transcoding, Transrating, Inband DTMF Interworking, and SRTP Encryption and Decryption, page 7-149
- Configuring Usage Limits for IPSec Encryption and Decryption and IPSec-Protected Signaling, page 7-153
- Example: Limiting Resource Usage, page 7-167

Configuring Resource Costs for Transcoding, Transrating, Inband DTMF Interworking, and SRTP Encryption and Decryption

The resource costs that you have configured are used to calculate and compare the total weighted resource usage against the maximum total usage that you have configured. Table 7-10 shows the default resource costs. You can modify these resource costs to suit the requirements of your operating environment.

This task explains how to configure resource costs for transcoding, transrating, inband DTMF interworking, and SRTP encryption and decryption.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**

4. **media-gateway policy type {default | local | {remote {ipv4 | ipv6}** *ip-address* **[port** *port-number*]}}
5. **transcode audio cost** *number*
6. **transcode video cost** *number*
7. **transrate audio cost** *number*
8. **interwork inband-dtmf cost** *number*
9. **interwork srtp cost** *number*
10. **end**
11. **show sbc** *sbc-name* **sbe media-gateway-policy**
12. **show sbc** *sbc-name* **sbe media-gateway-policy [stats | type {default | local | remote {ipv4 | ipv6}** *ip-address* **[port** *port-number*]}]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mySbc`	Enters the SBC service mode. • *sbc-name*—Name of the SBC.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the SBE configuration mode.

	Command or Action	Purpose
Step 4	**media-gateway policy type {default | local | {remote {ipv4 | ipv6} *ip-address* [port *port-number*]}}** **Example:** `Router(config-sbc-sbe)# media-gateway policy type remote ipv4 192.0.2.26 6886`	Configures a media gateway policy. • **default**—Specifies that the media gateway policy must be applied to all media gateways configured on the SBC. A default media gateway policy is applied on a media gateway (local or remote) when no other media policy is applied on the media gateway. • **local**—Specifies that the media gateway policy must be applied to the media gateway that is locally configured on the SBC. • **remote**—Specifies that the media gateway policy must be applied to a remote media gateway. • **ipv4**—Specifies that the remote media gateway has an IPv4 IP address. • **ipv6**—Specifies that the remote media gateway has an IPv6 IP address. • *ip-address*— IP address of the remote media gateway. The IP address can be in the IPv4 format or IPv6 format. • **port**—Specifies the port number of the remote media gateway. • *port-number*—Port number of the remote media gateway. Enters the media policy configuration mode.
Step 5	**transcode audio cost** *number* **Example:** `Router(config-sbc-sbe-media-pol)# transcode audio cost 10`	Specifies the resource cost for transcoding an audio stream. • *number*—Resource cost. The range is from 1 to 4294967295. As mentioned in Table 7-10, the default cost is 10.
Step 6	**transcode video cost** *number* **Example:** `Router(config-sbc-sbe-media-pol)# transcode video cost 55`	Specifies the resource cost for transcoding a video stream. • *number*—Resource cost. The range is from 1 to 4294967295. As mentioned in Table 7-10, the default cost is 50.
Step 7	**transrate audio cost** *number* **Example:** `Router(config-sbc-sbe-media-pol)# transrate audio cost 10`	Specifies the resource cost for transrating an audio stream. • *number*—Resource cost. The range is from 1 to 4294967295. As mentioned in Table 7-10, the default cost is 6.
Step 8	**interwork inband-dtmf cost** *number* **Example:** `Router(config-sbc-sbe-media-pol)# interwork inband-dtmf cost 6`	Specifies the resource cost for an audio stream using inband DTMF interworking. • *number*—Resource cost. The range is from 1 to 4294967295. As mentioned in Table 7-10, the default cost is 4.

	Command or Action	Purpose
Step 9	**interwork srtp cost** *number* **Example:** `Router(config-sbc-sbe-media-pol)# interwork srtp cost 15`	Specifies the resource cost for an audio or video stream using SRTP encryption and decryption. • *number*—Resource cost. The range is from 1 to 4294967295. As mentioned in Table 7-10, the default cost is 15.
Step 10	**end** **Example:** `Router(config-sbc-sbe-media-pol)# end`	Exits the media policy configuration mode, and enters the privileged EXEC mode.
Step 11	**show sbc** *sbc-name* **sbe media-gateway-policy** **Example:** `Router# show sbc mySbc sbe media-gateway-policy`	Displays the details of all media gateway policies. • *sbc-name*—Name of the SBC.
Step 12	**show sbc** *sbc-name* **sbe media-gateway-policy [stats | type {default | local | remote {ipv4 | ipv6}** *ip-address* **[port** *port-number***]}]** **Example:** `Router# show sbc mySbc sbe media-gateway-policy type remote ipv4 192.0.2.26 port 6886`	Displays the details of the specified media gateway policy. • *sbc-name*—Name of the SBC. • *ip-address*— IP address of the remote media gateway. The IP address can be in the IPv4 format or IPv6 format. • *port-number*—Port number of the remote media gateway.

The following example shows the output of the **show sbc sbe media-gateway-policy type** command for a specified media gateway policy:

```
Router# show sbc mySbc sbe media-gateway-policy type remote ipv4 192.0.2.26 port 6886

Gateway Policy Type                =     REMOTE
_____________________________________________________________
Remote vpn                         =     0
Remote address type                =     IPV4
Remote address                     =     192.0.2.26
Remote Port                        =     6886
Media Limit Table                  =
Transcode Audio Cost               =     10
Transrate Audio Cost               =     6
```

Configuring Usage Limits for Transcoding, Transrating, Inband DTMF Interworking, and SRTP Encryption and Decryption

This task describes how to configure usage limits for transcoding, transrating, inband DTMF interworking, and SRTP encryption and decryption.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **media-policy** *policy-name*
5. **type {cac-policy | gateway}**
6. **transcode audio maximum** *number*
7. **transcode video maximum** *number*
8. **transrate audio maximum** *number*
9. **interwork inband-dtmf maximum** *number*
10. **interwork srtp maximum** *number*
11. **total resource maximum** *number*
12. **exit**
13. **media-gateway policy type {default | local | {remote {ipv4 | ipv6}** *ip-address* **[port** *port-number***]}}**
14. **media limits** *policy-name*
15. **end**
16. **show sbc** *sbc-name* **sbe media-policy**
17. **show sbc** *sbc-name* **sbe media-policy** *policy-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mySbc`	Enters the SBC service mode. • *sbc-name*—Name of the SBC.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the SBE configuration mode.
Step 4	**media-policy** *policy-name* **Example:** `Router(config-sbc-sbe)# media-policy media_policy2`	Specifies the media policy to be created. • *policy-name*—Name of the media policy. Enters the media policy configuration mode.

	Command or Action	Purpose
Step 5	**type {cac-policy | gateway}** **Example:** `Router(config-sbc-sbe-media-pol)# type gateway`	Specifies the type of media policy table to be configured. You can specify one of the following media policy types: • **cac-policy**—Specifies that a media policy table must be configured for a CAC-policy type policy. • **gateway**—Specifies that a media policy table must be configured for a gateway type policy.
Step 6	**transcode audio maximum** *number* **Example:** `Router(config-sbc-sbe-media-pol)# transcode audio maximum 20000`	Specifies the maximum number of media streams that can be audio transcoded at any point of time. • *number*—Number of media streams. The range is from 1 to 4294967295. The default is 4294967295.
Step 7	**transcode video maximum** *number* **Example:** `Router(config-sbc-sbe-media-pol)# transcode video maximum 20000`	Specifies the maximum number of media streams that can be video transcoded at any point of time. • *number*—Number of media streams. The range is from 1 to 4294967295. The default is 4294967295.
Step 8	**transrate audio maximum** *number* **Example:** `Router(config-sbc-sbe-media-pol)# transrate audio maximum 6000`	Specifies the maximum number of media streams that can be audio transrated at any point of time. • *number*—Number of media streams. The range is from 1 to 4294967295. The default is 4294967295.
Step 9	**interwork inband-dtmf maximum** *number* **Example:** `Router(config-sbc-sbe-media-pol)# interwork inband-dtmf maximum 2000`	Specifies the maximum number of media streams that can use the inband DTMF interworking resource at any point of time. • *number*—Number of media streams. The range is from 1 to 4294967295. The default is 4294967295.
Step 10	**interwork srtp maximum** *number* **Example:** `Router(config-sbc-sbe-media-pol)# interwork srtp maximum 500`	Specifies the maximum number of media streams that can use the SRTP interworking resource at any point of time. • *number*—Number of media streams. The range is from 1 to 4294967295. The default is 4294967295.
Step 11	**total resource maximum** *number* **Example:** `Router(config-sbc-sbe-media-pol)# total resource maximum 35000`	Specifies the total number of video and audio streams that can use transcoding, transrating, inband DTMF interworking, and SRTP encryption and decryption—weighted by the costs assigned to each of these resources. • *number*—Number of media streams. The range is from 1 to 4294967295. The default is 4294967295.
Step 12	**exit** **Example:** `Router(config-sbc-sbe-media-pol)# exit`	Exits the SBE media policy configuration mode, and enters the SBE configuration mode.

	Command or Action	Purpose
Step 13	**media-gateway policy type {default | local | {remote {ipv4 | ipv6} *ip-address* [port *port-number*]}}** **Example:** `Router(config-sbc-sbe)# media-gateway policy type default`	Configures a media gateway policy. • **default**—Specifies that the media gateway policy must be applied to all media gateways configured on the SBC. A default media gateway policy is applied on a media gateway (local or remote) when no other media policy is applied on the media gateway. • **local**—Specifies that the media gateway policy must be applied to the media gateway that is locally configured on the SBC. • **remote**—Specifies that the media gateway policy must be applied to a remote media gateway. • **ipv4**—Specifies that the remote media gateway has an IPv4 IP address. • **ipv6**—Specifies that the remote media gateway has an IPv6 IP address. • *ip-address*—IP address of the remote media gateway. The IP address can be in the IPv4 format or IPv6 format. • **port**—Specifies the port number of the remote media gateway. • *port-number*—Port number of the remote media gateway. Enters the media policy configuration mode.
Step 14	**media limits** *policy-name* **Example:** `Router(config-sbc-sbe-mg-pol)# media limits media_policy2`	Specifies the media policy to be associated with the CAC policy table entry or applied on the media gateway. • *policy-name*—Name of the media policy.
Step 15	**end** **Example:** `Router(config-sbc-sbe-media-pol)# end`	Exits the media policy configuration mode, and enters the privileged EXEC mode.
Step 16	**show sbc** *sbc-name* **sbe media-policy** **Example:** `Router# show sbc mySbc sbe media-policy`	Displays details of all media policies. These details include the resource usage limits that you have configured. • *sbc-name*—Name of the SBC service.
Step 17	**show sbc** *sbc-name* **sbe media-policy** *policy-name* **Example:** `Router# show sbc mySbc sbe media-policy`	Displays details of all media policies. These details include the resource usage limits that you have configured. • *sbc-name*—Name of the SBC service. • *policy-name*—Name of the media policy.

The following example shows the output of the **show sbc sbe media-policy** command for a specified media policy:

```
Router# show sbc mySbc sbe media-policy my_media_policy

Policy Name:  my_media_policy
_____________________________________________________________
Type                                        = gateway
Audio transcode limit                       = 30
Audio transrate limit                       = 30
Video transcode limit                       = 30
Inband-dtmf-iw  limit                       = 10
SRTP-iw         limit                       = 20
Total resource  limit                       = 40
```

Configuring Usage Limits for IPSec Encryption and Decryption and IPSec-Protected Signaling

This task explains how to configure usage limits for IPSec encryption and decryption and IPSec-protected signaling.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **cac-policy-set** {*policy-set-id* | **copy** {**source** *policy-set-id* **destination** *policy-set-id*} | **swap** {**source** *policy-set-id* **destination** *policy-set-id*} | **averaging-period** {*average-number average-period*}
5. **cac-table** *table-name*
6. **entry** *entry-id*
7. **ipsec maximum registers** *number*
8. **ipsec maximum calls** *number*
9. **end**
10. **show sbc** *sbc-name* **sbe cac-policy-set** *policy-set-id* **detail**
11. **show sbc** *sbc-name* **sbe cac-policy-set** *policy-set-id* **table** *table-name* **detail**
12. **show sbc** *sbc-name* **sbe cac-policy-set** *policy-set-id* **table** *table-name* **entry** *entry-id*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mySbc`	Enters the SBC service mode. • *sbc-name*—Name of the SBC.
Step 3	**sbe** **Example:** `Router(config-sbc)# sbe`	Enters the SBE configuration mode.
Step 4	**cac-policy-set {***policy-set-id* **| copy {source** *policy-set-id* **destination** *policy-set-id***} | swap {source** *policy-set-id* **destination** *policy-set-id***} | averaging-period {average-number** *average-period***}** **Example:** `Router(config-sbc-sbe)# cac-policy-set 1`	Enters the CAC policy set configuration mode within an SBE entity. If the policy set does not exist, it is created. • *policy-set-id*—CAC policy set number. The range is from 1 to 2147483647. **Note** The keywords and arguments of the **cac-policy-set** command that are not relevant to this section have not been described here. See *Cisco Unified Border Element (SP Edition) Command Reference: Unified Model* for information about these keywords and arguments.
Step 5	**cac-table** *table-name* **Example:** `Router(config-sbc-sbe-cacpolicy)# cac-table t1`	Enters the CAC table configuration mode within an SBE policy set. If the CAC table does not exist, it is created. • *table-name*—Name of the CAC table.
Step 6	**entry** *entry-id* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable)# entry 1`	Enters the mode for configuring an entry in a CAC table. If the entry does not exist, it is created. • *entry-id*—ID of the CAC table entry.
Step 7	**ipsec maximum registers** *number* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# ipsec maximum registers 10`	Specifies the maximum number of endpoint registrations that can use IPsec encryption and decryption on their signaling link to the SBC. • *number*—Number of endpoint registrations. The range is from 1 to 4294967295. The default is 4294967295. **Note** This configuration is not used when determining the call scope. In addition, this configuration is not used when the SBC performs the Interconnection Border Control Function (IBCF) because all registrations are stateless and the SBC cannot determine whether a registration is new.

	Command or Action	Purpose
Step 8	**ipsec maximum calls** *number* **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# ipsec maximum calls 5`	Specifies the maximum number of calls that can use IPsec-protected signaling. • *number*—Number of calls. The range is from 1 to 4294967295. The default is 4294967295.
Step 9	**end** **Example:** `Router(config-sbc-sbe-cacpolicy-cactable-entry)# end`	Exits the SBE mode, and returns to the privileged EXEC mode.
Step 10	**show sbc** *sbc-name* **sbe cac-policy-set** *policy-set-id* **detail** **Example:** `Router(config)# show sbc mySbc sbe cac-policy-set 1 detail`	Shows detailed information pertaining to a CAC policy set. • *sbc-name*—Name of the SBC. • *policy-set-id*—ID of the policy set.
Step 11	**show sbc** *sbc-name* **sbe cac-policy-set** *policy-set-id* **table** *table-name* **detail** **Example:** `Router(config)# show sbc mySbc sbe cac-policy-set 1 table t1 detail`	Shows detailed information pertaining to a table in a CAC policy set. • *sbc-name*—Name of the SBC. • *policy-set-id*—ID of the policy set. • *table-name*—Name of the table.
Step 12	**show sbc** *sbc-name* **sbe cac-policy-set** *policy-set-id* **table** *table-name* **entry** *entry-id* **Example:** `Router(config)# show sbc mySbc sbe cac-policy-set 1 table t1 entry 1`	Shows detailed information pertaining to an entry in a table in a CAC policy set. • *sbc-name*—Name of the SBC. • *policy-set-id*—ID of the policy set. • *table-name*—Name of the table. • *entry-id*—ID of the CAC table entry.

The following example shows the output of the **show sbc sbe cac-policy-set table entry** command:

```
Router# show sbc mySbc sbe cac-policy-set 1 table t1 entry 1
SBC Service "mySbc"
CAC Averaging period 1: 60 sec
CAC Averaging period 2: 0 sec

CAC Policy Set 1
  Active policy set: No
  Description:
  First CAC table:
  First CAC scope: global

  Table name: t1
    Description:
    Table type: policy-set
    Total call setup failures (due to non-media limits): 0

    Entry 1
    CAC scope:
    CAC scope prefix length: 0
    Action: Not set
```

```
    Number of call setup failures (due to non-media limits): 0
.
.
.
media bandwidth policing:        Degrade
Caller ptime:                    None (default)
Callee ptime:                    None (default)
Caller inband DTMF mode:         Inherit(default)
Callee inband DTMF mode:         Inherit(default)
Media policy limit table name:   mp1
IPsec maximum registers:         10
IPsec maximum calls:             5
```

Configuration Examples for Implementing Policies

This section provides the following configuration examples:

Example: Implementing Number Analysis

The following example shows call processing handled with number analysis working with a category routing table in the following manner: 1) shows number analysis, based on number categorization, of a set of dialed digits to determine which is a valid telephone number, 2) shows how the categorized calls are handled with a call routing policy based on category, and 3) shows source address manipulation.

This task configures text address validation and source address manipulation for a number analysis table.

Under 1) for any new call, the SBC inspects the first few digits of the called number that is determined by the “match-prefix” and categorizes the call, based on the category configured under the “na-dst-prefix-table Determine-Category” entry. For example, calls with a prefix of 911 in the destination number are categorized as EMERGENCY calls; calls with a prefix of 919 are Legit_Call, and calls with a prefix of 900 are Blocked_Number calls.

Under 2) routing policy is defined based on category as specified by the “rtg-category-table Category_Routing” table that allows EMERGENCY calls and Legit_Call and rejects all Blocked_Number calls.

```
call-policy-set 1
  first-inbound-na-table Determine-Category
  first-call-routing-table Category_Routing
  rtg-src-adjacency-table Routing-Table-2
    entry 1
      action complete
      dst-adjacency Adj-502
```

```
    match-adjacency Adj-503
  entry 2
    action complete
    dst-adjacency Adj-503
    match-adjacency Adj-502
rtg-category-table Category_Routing ==> 2) categorized calls handled with routing policy
  entry 1
    action next-table Routing-Table-1
    match-category EMERGENCY
  entry 2
    action next-table Routing-Table-2
    match-category Legit_Call
  entry 3
    action complete
    match-category Blocked_Number
rtg-src-adjacency-table Routing-Table-1
  entry 1
    action complete
    dst-adjacency Adj-502
    match-adjacency Adj-501
  entry 2
    action complete
    dst-adjacency Adj-501
    match-adjacency Adj-502
na-dst-prefix-table Determine-Category =====> 1) number analysis based on categorization
  entry 1
    action accept
    category EMERGENCY
    match-prefix 911
  entry 2
    action accept
    category Legit_Call
    match-prefix 919
  entry 3
    action reject
    category Blocked_Number
    match-prefix 900
 na-dst-address-table mytable
     entry 1
       action accept
       edit-src del-prefix 3 =====> 3) source address manipulation
       match-address 123456 digits
     entry 2
       action accept
       edit-src del-suffix 1
       match-address ^.* regex
```

Example: Configuring Administrative Domain

The following example shows how to configure the administrative domains:

```
adjacency sip SIPP1A
    admin-domain SIPP1A
    inherit profile preset-access
    signaling-address ipv4 10.10.100.140
    statistics method summary
    signaling-port 7065
    remote-address ipv4 10.10.100.11 255.255.255.255
    signaling-peer 10.10.100.11
    signaling-peer-port 7065
    registration rewrite-register
    attach
```

```
adjacency sip SIPP1B
 admin-domain SIPP1B
 inherit profile preset-access
 signaling-address ipv4 10.10.100.140
 statistics method summary
 signaling-port 7066
 remote-address ipv4 10.10.100.12 255.255.255.255
 signaling-peer 10.10.100.12
 signaling-peer-port 7066
 registration rewrite-register
 attach
adjacency sip Registrar
 inherit profile preset-core
 signaling-address ipv4 10.10.100.140
 statistics method summary
 signaling-port 7020 7029
 remote-address ipv4 10.10.100.12 255.255.255.255
 signaling-peer 10.10.100.12
 signaling-peer-port 7068
 registration contact username passthrough
 registration target address 10.10.100.12
 registration target port 7069
 attach
cac-policy-set averaging-period 1 120
cac-policy-set averaging-period 2 40
cac-policy-set 10
 first-cac-table TAB1
 first-cac-scope src-adjacency
 cac-table TAB1
  table-type limit adjacency
  entry 1
   match-value SIPP1A
 .
 .
 .
   action cac-complete
  complete
cac-policy-set 20
 first-cac-table TAB1
 cac-table TAB1
  table-type policy-set
  entry 1
   max-call-rate-per-scope 600 averaging-period 1
   action cac-complete
  complete
cac-policy-set global 20
call-policy-set 10
 first-call-routing-table RTG_TBL
 first-reg-routing-table REG_TBL
 rtg-src-adjacency-table RTG_TBL
  entry 1
   match-adjacency SIPP1A
   dst-adjacency SIPP1B
   action complete
 rtg-src-adjacency-table REG_TBL
  entry 1
   match-adjacency SIPP1A
   dst-adjacency Registrar
   action complete
 complete
call-policy-set 20
 first-call-routing-table RTG_TBL
 first-reg-routing-table REG_TBL
 rtg-src-adjacency-table RTG_TBL
```

```
     entry 1
      match-adjacency SIPP1A
      dst-adjacency SIPP1B
      action complete
     entry 2
      match-adjacency SIPP1B
      dst-adjacency SIPP1A
      action complete
    rtg-src-adjacency-table REG_TBL
     entry 1
      match-adjacency SIPP1A
      dst-adjacency Registrar
      action complete
     entry 2
      match-adjacency SIPP1B
      dst-adjacency Registrar
      action complete
    complete
   call-policy-set default 20
   admin-domain SIPP1A
    cac-policy-set 10
    call-policy-set 10
! no admin-domain for SIPP1B defaults to default call-policy
```

Example: Implementing Call Admission Control Policy Sets and CAC Tables

The following example shows how to configure call admission control policy sets and CAC tables:

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# cac-policy-set 1
Router(config-sbc-sbe-cacpolicy)# first-cac-scope global
Router(config-sbc-sbe-cacpolicy)# first-cac-table STANDARD-LIST-BY-ACCOUNT
Router(config-sbc-sbe-cacpolicy)# cac-table STANDARD-LIST-BY-ACCOUNT
Router(config-sbc-sbe-cacpolicy-cactable)# table-type limit dst-account
Router(config-sbc-sbe-cacpolicy-cactable)# entry 1
Router(config-sbc-sbe-cacpolicy-cactable-entry)# match-value SIP-CUSTOMER-1
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-num-calls 100
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-call-rate-per-scope 20
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-bandwidth 1000000 bps
Router(config-sbc-sbe-cacpolicy-cactable-entry)# callee-privacy never
Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete
Router(config-sbc-sbe-cacpolicy-cactable-entry)# exit
Router(config-sbc-sbe-cacpolicy-cactable)# entry 2
Router(config-sbc-sbe-cacpolicy-cactable-entry)# match-value SIP-CUSTOMER-2
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-num-calls 100
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-call-rate-per-scope 20
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-bandwidth 1000000 bps
Router(config-sbc-sbe-cacpolicy-cactable-entry)# transcode deny
Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete
Router(config-sbc-sbe-cacpolicy-cactable-entry)# exit
Router(config-sbc-sbe-cacpolicy-cactable)# exit
Router(config-sbc-sbe-cacpolicy)# complete
```

The following example limits the total number of concurrent calls per SBC (global limit) to 2000 and the number of concurrent calls per source adjacency to 5. If an adjacency has 5 calls that are active, it is not allowed to make the sixth call even if the total number of active calls on the SBC is less than 2000. Also, if the total number of active calls on the SBC is 2000, an adjacency is not allowed to make a call even if it has no active calls.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# cac-policy-set 1
Router(config-sbc-sbe-cacpolicy)# first-cac-scope global
Router(config-sbc-sbe-cacpolicy)# first-cac-table first_policy_table
Router(config-sbc-sbe-cacpolicy)# cac-table first_policy_table
Router(config-sbc-sbe-cacpolicy-cactable)# table-type policy-set
Router(config-sbc-sbe-cacpolicy-cactable)# entry 1
Router(config-sbc-sbe-cacpolicy-cactable-entry)# cac-scope src-adjacency
Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-num-calls 5
Router(config-sbc-sbe-cacpolicy-cactable-entry)# exit
Router(config-sbc-sbe-cacpolicy-cactable)# entry 2
Router(config-sbc-sbe-cacpolicy-cactable-entry)# cac-scope global
Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-num-calls 2000
Router(config-sbc-sbe-cacpolicy-cactable-entry)# exit
Router(config-sbc-sbe-cacpolicy-cactable)# exit
Router(config-sbc-sbe-cacpolicy)# complete
Router(config-sbc-sbe-cacpolicy)# exit
Router(config-sbc-sbe)# cac-policy-set global 1
```

The following example limits the number of concurrent calls per subscriber to 5 with no global limit:

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# cac-policy-set 1
Router(config-sbc-sbe-cacpolicy)# first-cac-table first_policy_table
Router(config-sbc-sbe-cacpolicy)# cac-table first_policy_table
Router(config-sbc-sbe-cacpolicy-cactable)# table-type policy-set
Router(config-sbc-sbe-cacpolicy-cactable)# entry 1
Router(config-sbc-sbe-cacpolicy-cactable-entry)# cac-scope src-adjacency
Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-num-calls 5
Router(config-sbc-sbe-cacpolicy-cactable-entry)# exit
Router(config-sbc-sbe-cacpolicy-cactable)# exit
Router(config-sbc-sbe-cacpolicy)# complete
Router(config-sbc-sbe-cacpolicy)# exit
Router(config-sbc-sbe)# cac-policy-set global 1
```

You could also achieve this with the following configuration:

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# cac-policy-set 3
Router(config-sbc-sbe-cacpolicy)# first-cac-table 1
Router(config-sbc-sbe-cacpolicy)# first-cac-scope src-adjacency
Router(config-sbc-sbe-cacpolicy)# cac-table 1
Router(config-sbc-sbe-cacpolicy-cactable)# table-type limit all
Router(config-sbc-sbe-cacpolicy-cactable)# entry 1
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-num-calls 5
Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete
Router(config-sbc-sbe-cacpolicy-cactable-entry)# exit
Router(config-sbc-sbe-cacpolicy-cactable)# exit
Router(config-sbc-sbe-cacpolicy)# complete
Router(config-sbc-sbe-cacpolicy)# exit
Router(config-sbc-sbe)# cac-policy-set global 1
```

Both of the above configurations will limit the number of concurrent calls per subscriber to 5. There is no global limit.

In the following example, if the bandwidth used by an adjacency whose source IP address is 1.1.1.1 is less than 1 Mbps, then the call is admitted. Also adjacencies with a source IP address of 2.2.2.2 that use less than 2 Mbps of bandwidth will have their calls admitted.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# cac-policy-set 1
Router(config-sbc-sbe-cacpolicy)# first-cac-scope global
Router(config-sbc-sbe-cacpolicy)# first-cac-table first_policy_table
Router(config-sbc-sbe-cacpolicy)# cac-table first_policy_table
Router(config-sbc-sbe-cacpolicy-cactable)# table-type limit src-adjacency
Router(config-sbc-sbe-cacpolicy-cactable)# entry 1
Router(config-sbc-sbe-cacpolicy-cactable-entry)# match-value 1.1.1.1
Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-bandwidth 1 Mbps
Router(config-sbc-sbe-cacpolicy-cactable-entry)# exit
Router(config-sbc-sbe-cacpolicy-cactable)# entry 2
Router(config-sbc-sbe-cacpolicy-cactable-entry)# match-value 2.2.2.2
Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-bandwidth 2 Mbps
Router(config-sbc-sbe-cacpolicy-cactable-entry)# exit
Router(config-sbc-sbe-cacpolicy-cactable)# exit
Router(config-sbc-sbe-cacpolicy)# complete
Router(config-sbc-sbe-cacpolicy)# exit
Router(config-sbc-sbe)# cac-policy-set global 1
```

This example allows 10 calls, 100 updates, a max-in-call-msg-rate and a max-out-call-msg-rate of 5000 msg/min for any source adjacency:

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# cac-policy-set 1
Router(config-sbc-sbe-cacpolicy)# first-cac-scope sub-category
Router(config-sbc-sbe-cacpolicy)# first-cac-table first_policy_table
Router(config-sbc-sbe-cacpolicy)# cac-table first_policy_table
Router(config-sbc-sbe-cacpolicy-cactable)# table-type policy-set
Router(config-sbc-sbe-cacpolicy-cactable)# entry 1
Router(config-sbc-sbe-cacpolicy-cactable-entry)# cac-scope src-adjacency
Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-num-calls 10
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-updates 100
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-in-call-msg-rate 5000
Router(config-sbc-sbe-cacpolicy-cactable-entry)# max-out-call-msg-rate 5000
Router(config-sbc-sbe-cacpolicy-cactable-entry)# exit
Router(config-sbc-sbe-cacpolicy-cactable)# exit
Router(config-sbc-sbe-cacpolicy)# complete
Router(config-sbc-sbe-cacpolicy)# exit
Router(config-sbc-sbe)# cac-policy-set global 1
```

Example: Multiple SBC Media Bypass

The following example shows how to configure a media bypass across two or more SBCs as shown in Figure 7-7, when making calls from endpoint 1 to endpoint 2. In the example, the adjacencies configured on each SBC is named *access*, for endpoint facing adjacency, and *core* for proxy facing adjacency. To achieve media bypass for calls from endpoint 2 to endpoint 1, two CAC entries with match-value as *access* and *core* must be configured with the same settings in the CAC table.

SBC 1:

```
sbc SBC1
   sbe
      adjacency sip access
        .
        .
        .
        media bypass tag 1 enterprise1
        .
        .
        .
      adjacency sip core
        .
        .
        .
      cac-policy-set 1
         cac-table MyTable
           table-type limit src-adjacency
             entry 1
            .
            .
            .
                match-value access
               media bypass type full hairpin
               caller media bypass enable
               callee media bypass enable
               action cac-complete
           entry 2
               session-refresh renegotiation suppress
            .
            .
            .
```

SBC 2:

```
sbc SBC2
   sbe
      adjacency sip access
        .
        .
        .
        media bypass tag 1 enterprise1
        .
        .
        .
      adjacency sip core
        .
        .
        .
      cac-policy-set 1
         cac-table MyTable
           table-type limit src-adjacency
             entry 1
            .
            .
            .
       match-value core
                media bypass type full hairpin
                caller media bypass enable
                callee media bypass enable
                action cac-complete
```

The following example shows the output of the **show sbc sbe adjacencies detail** command:

```
Router# show sbc SBC1 sbe adjacencies access detail

SBC Service SBC1
  Adjacency access (SIP)

      Media Bypass Tag List:
       Tag 1:                    tag1
       Tag 2:                    tag2
  Media Bypass Max Out Data Length:        1024
```

The following example shows the output of the **show sbc sbe cac-policy-set table entry detail** command:

```
Router# show sbc SBC1 sbe cac-policy-set 1 table MyTable entry 1 detail

SBC Service "SBC1"

    CAC Policy Set 1
        Active policy set: No
        Description:
        Averaging period: 60 sec
        First CAC table:
        First CAC scope: global

        Table name: MyTable
        Description:
        Table type: policy-set

        Entry 1
        Action: CAC Complete
        …
        Media Bypass Type: Full Partial
        Caller Media Bypass: Enabled
        Callee Media Bypass: Enabled
```

Example: Configuring Hunting

The following example shows how to hunt for other routes or destination adjacencies in case of a failure in a SIP mode:

```
Router# configure terminal
Router(config)# sbc mySBC
Router(config-sbc)# sbe
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# first-call-routing-table SAMPLE
Router(config-sbc-sbe-rtgpolicy)# first-reg-routing-table SAMPLE
Router(config-sbc-sbe-rtgpolicy)# rtg-src-adjacency-table SAMPLE
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency TA1
Router(config-sbc-sbe-rtgpolicy-entry)# match-adjacency Hunted
Router(config-sbc-sbe-rtgpolicy-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency TA2
Router(config-sbc-sbe-rtgpolicy-entry)# match-adjacency Hunted
Router(config-sbc-sbe-rtgpolicy-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 3
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
```

```
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency TA3
Router(config-sbc-sbe-rtgpolicy-entry)# match-adjacency Hunted
Router(config-sbc-sbe-rtgpolicy-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 4
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency TA4
Router(config-sbc-sbe-rtgpolicy-entry)# match-adjacency Hunted
Router(config-sbc-sbe-rtgpolicy-entry)# exit
Router(config-sbc-sbe-rtgpolicy)# complete
Router(config-sbc-sbe-rtgpolicy)# exit
Router(config-sbc-sbe)# adjacency sip Hunted
Router(config-sbc-sbe-adj-sip)# hunting-trigger 403 415 503 604
Router(config-sbc-sbe-adj-sip)# exit
```

The following example shows how to configure Cisco Unified Border Element (SP Edition) to hunt for other H.323 routes or destination adjacencies in case of a failure:

```
Router# configure terminal
Router(config)# sbc mySBC
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency h323 adjacency-name
Router(config-sbc-sbe-h323)# hunting-trigger noBandwidth
Router(config-sbc-sbe-h323)# hunting-trigger unreachableDestination
Router(config-sbc-sbe-h323)# hunting-mode altEndps
Router(config-sbc-sbe-h323)# exit
```

Example: Allowing Asymmetric Payload Types

The following example shows how to configure the SBC to specify support for Asymmetric payload types on the mySBC SBC:

```
Router# configure terminal
Router(config)# sbc mySBC
Router(config-sbc)# sbe
Router(config-sbc-sbe)# cac-policy-set 1
Router(config-sbc-sbe-cacpolicy)# first-cac-table my_table
Router(config-sbc-sbe-cacpolicy)# cac-table TAB1
Router(config-sbc-sbe-cacpolicy-cactable)# table-type policy-set
Router(config-sbc-sbe-cacpolicy-cactable)# entry 1
Router(config-sbc-sbe-cacpolicy-cactable-entry)# action cac-complete
Router(config-sbc-sbe-cacpolicy-cactable-entry)# payload-type asymmetric allowed
Router(config-sbc-sbe-cacpolicy-cactable-entry)# complete
Router(config-sbc-sbe)# cac-policy-set global 1
Router(config-sbc-sbe)# end
Router#
```

The following example shows a SIP/SIP call with a single CAC Policy Set table allowing Asymmetric payload types:

Configuration:

```
cac-policy-set 1
 first-cac-table TAB1
 cac-table TAB1
  table-type policy-set
  entry 1
   payload-type asymmetric allowed
   action cac-complete
 complete
cac-policy-set global 1
```

Call succeeds with the following invite, and 200 messages exchanged:

Invite Sent:

```
2010-01-12 16:28:35
UDP message sent:

INVITE sip:service@2.0.0.5:5078 SIP/2.0
Via: SIP/2.0/UDP 2.0.0.3:5078;branch=z9hG4bK-32567-1-0
From: sipp ;tag=32567SIPpTag091
To: sut
Call-ID: 1-32567@2.0.0.3
CSeq: 1 INVITE
Contact: sip:sipp@2.0.0.3:5078
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Length:  127

v=0
o=user1 53655765 2353687637 IN IP4 2.0.0.3
s=-
c=IN IP4 2.0.0.3
t=0 0
m=audio 6000 RTP/AVP 18
a=rtpmap:18 G729/8000
```

200 Received:

```
2010-01-12 16:28:35
UDP message received [485] bytes :

SIP/2.0 200 OK
Call-ID: 1-32567@2.0.0.3
CSeq: 1 INVITE
From: sipp ;tag=32567SIPpTag091
To: sut ;tag=sip+1+1060000+47e93fd7
Via: SIP/2.0/UDP 2.0.0.3:5078;branch=z9hG4bK-32567-1-0
Server: CISCO-SBC/2.x
Content-Length: 146
Contact:
Content-Type: application/sdp

v=0
o=user1 5338645241744 5338645241744 IN IP4 10.10.20.20
s=-
c=IN IP4 10.10.20.20
t=0 0
m=audio 16384 RTP/AVP 118
a=rtpmap:118 G729/8000
```

The following example shows a SIP/SIP call with a single CAC policy set table disallowing Asymmetric payload types:

Configuration:

```
cac-policy-set 1
 first-cac-table TAB1
 cac-table TAB1
  table-type policy-set
  entry 1
   payload-type asymmetric disallowed
   action cac-complete
```

```
    complete
   cac-policy-set global 1
```

Call fails with the following invite and error messages:

Invite Sent:

```
2010-01-12 16:39:09
UDP message sent:

INVITE sip:service@2.0.0.5:5078 SIP/2.0
Via: SIP/2.0/UDP 2.0.0.3:5078;branch=z9hG4bK-32584-1-0
From: sipp ;tag=32584SIPpTag091
To: sut
Call-ID: 1-32584@2.0.0.3
CSeq: 1 INVITE
Contact: sip:sipp@2.0.0.3:5078
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Length:  127

v=0
o=user1 53655765 2353687637 IN IP4 2.0.0.3
s=-
c=IN IP4 2.0.0.3
t=0 0
m=audio 6000 RTP/AVP 18
a=rtpmap:18 G729/8000
```

Error Message:

```
---------------------------------------------
Unexpected UDP message received:

SIP/2.0 400 Bad Request
Call-ID: 1-32584@2.0.0.3
CSeq: 1 INVITE
From: sipp ;tag=32584SIPpTag091
To: sut ;tag=sip+1+10b0000+3621b373
Via: SIP/2.0/UDP 2.0.0.3:5078;branch=z9hG4bK-32584-1-0
Server: CISCO-SBC/2.x
Content-Length: 0
Contact:
```

Example: Common IP Address Media Bypass

The following example shows how to configure the Common IP Address Media Bypass feature on the access-side-1 adjacency:

```
Router# configure terminal
Router(config)# sbc mySBC
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip access-side-1
Router(config-sbc-sbe-adj-sip)# media bypass auto-nat-tag-gen
Router(config-sbc-sbe-adj-sip)# end
Router# show sbc mySBC sbe adjacencies access-side-1 detail

SBC Service "mySBC "
  Adjacency access-side-1 (SIP)
    Status:                Detached
```

```
.
.
.
                Register unencrypted convert: Disabled
    Warrant Match-Order:       None
                Media Bypass Max Out Data Length:   1000
    Auto bypass NAT: Enabled
```

Example: Limiting Resource Usage

This section describes examples related to implementing the Limiting Resource Usage feature.

In the following example, the local media gateway is configured to support up to 1000 audio transcoded streams.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# media-policy audio_limit1
Router(config-sbc-sbe-media-pol)# type gateway
Router(config-sbc-sbe-media-pol)# transcode audio maximum 1000
Router(config-sbc-sbe-media-pol)# exit
Router(config-sbc-sbe)# media-gateway policy type local
Router(config-sbc-sbe-mg-pol)# media limits audio_limit1
```

In the following example, the remote media gateway at 192.0.2.26 is configured to support up to 1500 audio transcoded streams.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# media-policy audio_limit2
Router(config-sbc-sbe-media-pol)# type gateway
Router(config-sbc-sbe-media-pol)# transcode audio maximum 1500
Router(config-sbc-sbe-media-pol)# exit
Router(config-sbc-sbe)# media-gateway policy type remote ipv4 192.0.2.26 port 2000
Router(config-sbc-sbe-mg-pol)# media limits audio_limit2
```

In the following example, a default media gateway policy is configured to enable media gateways to support up to 2000 audio transcoded streams. This default media gateway policy is applied on a media gateway (local or remote) when no other media policy is applied on the media gateway.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# media-policy audio_limit3
Router(config-sbc-sbe-media-pol)# type gateway
Router(config-sbc-sbe-media-pol)# transcode audio maximum 2000
Router(config-sbc-sbe-media-pol)# exit
Router(config-sbc-sbe)# media-gateway policy type default
Router(config-sbc-sbe-mg-pol)# media limits audio_limit3
```

In the following example, a CAC policy is configured to restrict all destination numbers other than 911 to at most 5 media streams on which audio transcoding or audio transrating can be performed. Note that the CAC table commands to apply this restriction to all numbers other than 911 have not been included in this sequence of commands.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# media-policy media_streams_limit1
Router(config-sbc-sbe-media-pol)# type cac-policy
```

```
Router(config-sbc-sbe-media-pol)# transcode audio maximum 5
Router(config-sbc-sbe-media-pol)# transrate audio maximum 5
Router(config-sbc-sbe-media-pol)# exit
Router(config-sbc-sbe)# cac-policy-set 22
Router(config-sbc-sbe-cacpolicy)# cac-table table1
Router(config-sbc-sbe-cacpolicy-cactable)# entry 1
.
.
.
Router(config-sbc-sbe-cacpolicy-cactable-entry)# media limits media_streams_limit1
Router(config-sbc-sbe-cacpolicy-cactable-entry)# exit
Router(config-sbc-sbe-cacpolicy)# complete
```

Example: Configuration the CAC Threshold

The following example shows how to configure a charge of 10 per session and a call admission limit of 50, which allows 5 calls per second (50/10) through the system:

```
Router(config)# call admission new-model
Router(config)# call admission limit 50
Router(config)# call admission pppoe 10 1
```

Configuration Examples for Implementing Call Routing

This section provides the following configuration examples:

- Example: Routing with No Load Balancing, page 7-168
- Example: Least Cost Routing, page 7-169
- Example: Weighted Routing, page 7-170
- Example: Time-Based Routing, page 7-170
- Example: Regular Expression Based Routing, page 7-174
- Example: Trunk-Group ID Routing, page 7-174

Example: Routing with No Load Balancing

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# first-call-routing-table start_routing
Router(config-sbc-sbe-rtgpolicy)# rtg-dst-address-table start_routing
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-address XXX
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# next-table internal_routing
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-address XXXX
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# next-table external_routing
Router(config-sbc-sbe-rtgpolicy)# rtg-src-adjacency-table internal_routing
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-address sip_to_foo
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency sip_to_foo
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
```

```
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-address sip_to_bar
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency sip_to_bar
Router(config-sbc-sbe-rtgpolicy)# rtg-dst-address-table external_routing
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-address 208111
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency sip_to_foo
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-address 208222
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency sip_to_bar
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 3
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-address X
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency sip_to_softswitch
```

Example: Least Cost Routing

The following example configures a routing table that matches on category and then for each entry routes the call to a different least-cost table to choose the adjacency.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# rtg-category-table 1
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-category internal
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action next-table least_int_cost
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-category external
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action next-table least_ext_cost
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# exit
Router(config-sbc-sbe-rtgpolicy)# rtg-least-cost-table least_int_cost
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# cost 10
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# cost 50
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# exit
Router(config-sbc-sbe-rtgpolicy)# rtg-least-cost-table least_ext_cost
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# cost 50
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj3
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# cost 100
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj4
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
```

Example: Weighted Routing

In the above example, no two entries in one table have the same cost, so the weight parameter is left at the default of 1. If two or more entries with equal cost exist, and are selected for routing, then calls are distributed based on the weight configured (weight being the relative weight of an entry with respect to the lowest weight in the table). For example, if there are three entries of equal cost and weights of entry1, entry2, and entry3 are 1, 2, and 4 respectively, entry2 will route twice the number of calls as entry1, and entry3 will route four times the number of calls as entry1.

In the following example, all calls are routed to entry 1 because it has the lowest cost. However if routing fails, the remaining three entries all have the same cost, so the weight parameters determine which entry is picked. 80% of calls will be routed to SipAdj2 by entry 2, and the remaining 20% will be evenly divided between SipAdj3 and SipAdj4 (weights of entry 3 and entry 4 are left at a default of 1).

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# rtg-least-cost-table table1
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# cost 10
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# cost 50
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# weight 8
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 3
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# cost 50
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj3
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 4
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# cost 50
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj4
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
```

Example: Time-Based Routing

The following example shows two entries, one that routes traffic to Adj1 at all times and a second with a higher precedence that routes traffic to Adj2 if the time is between 9 AM and 6 PM on a weekday. When the two time periods overlap, the one with the higher precedence is chosen.

The two times ranges in entry 1 and entry 2 overlap. In this case, a call made between 9 AM to 6 PM on weekdays matches on both the entries but entry 2 is preferred due to its higher precedence.

If multiple ranges are specified as in entry 2, the Cisco Unified Border Element (SP Edition) will match the entry only during the intersection of the ranges. For example, entry 2 matches calls made Monday through Friday between 9 AM to 6 PM. The range is not Monday 9 AM to Friday 6 PM.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# rtg-time-table table1
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
```

```
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time date yr 2006 2020 mon 1 12 day
1 31
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 5
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time dow 1 5
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time tod hr  9 17 min 0 59
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 10
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
```

The following example configures a rule that routes traffic through adjacency SipAdj1 at all times, and through SipAdj2 between Monday 9 AM and Friday 6 PM.

```
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# rtg-time-table table1
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time date yr 2006 2020 mon 1 12 day
1 31
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 5
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time dow 1 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time tod hr 9 23 min 0 59
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 10
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 3
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time dow 2 4
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 10
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 4
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time dow 5 5
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time tod hr 0 17 min 0 59
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 10
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
```

In the configuration above, entry 2, entry 3, and entry 4 together specify the range Monday 9:00 AM through Friday 6:00 PM. This could also be accomplished by having one route for the entire time Monday through Friday with separate ranges to divert traffic during nights as follows:

```
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# rtg-time-table table1
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time date yr 2006 2020 mon 1 12 day
1 31
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 5
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time dow 1 5
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 10
```

```
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 3
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time dow 1 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time tod hr 0 8 min 0 59
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 20
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 4
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time dow 5 5
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time tod hr 18 23 min 0 59
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 20
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
```

The following example shows how to configure a rule that would route traffic through adjacencies SipAdj1 and SipAdj2 on Monday and Wednesday, respectively, between 9 AM and 6 PM, and through SipAdj3 at all other times.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# rtg-time-table table1
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time date yr 2006 2020 mon 1 12 day
1 31
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 5
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj3
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time dow 1 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time tod hr 9 17 min 0 59
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 10
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 3
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time dow 3 3
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time tod hr 9 17 min 0 59
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 10
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
```

The following example shows how to configure a rule that would route traffic through adjacency SipAdj1 on Saturdays and Sundays between 01 Mar 2008 through 30 Mar 2009, and through SipAdj2 all other times.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# rtg-time-table table1
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time date yr 2006 2020 mon 1 12 day
1 31
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 5
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time date yr 2008 2009 mon 3 3 day
1 30
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time dow 6 7
```

```
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 10
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
```

The following example shows how to configure a rule that would route traffic through adjacency SipAdj1 between 10:00 PM and 6:00 AM from Friday to Monday, and through SipAdj2 otherwise.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# rtg-time-table table1
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time date yr 2006 2020 mon 1 12 day
1 31
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 5
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time dow 5 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time tod hr 22 6 min 0 59
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 10
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
```

Note Time and day of the week are wrapping ranges, so the minimum can be larger than the maximum. For example, a single routing entry with the ranges Friday through Monday and 22:00 through 06:00 will match before 6 AM and after 10 PM on Friday, Saturday, Sunday and Monday.

In the following example, a user has all his routers running GMT no matter where they were so that they can be synchronized. But one router in New York has a time-based routing table that routes traffic to SipAdj1 at all times apart from Monday through Friday from 9 AM to 6 PM when it routes traffic to SipAdj2. The user wants these match times to refer to local time so it is necessary enter a **time-offset** command (New York is five hours behind GMT) as shown in the example below.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# time-offset hour 5 min 0 negative
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# rtg-time-table table1
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# use-time-offset
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time date yr 2006 2020 mon 1 12 day
1 31
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 5
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# use-time-offset
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time tod hr 9 17 min 0 59
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-time dow 6 7
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# precedence 10
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SipAdj2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
```

Example: Regular Expression Based Routing

The following example shows how to configure the regular expression based routing to match the user name or domain part of a source or destination SIP URI.

```
Router# configure terminal
Router(config)# sbc mySbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# rtg-dst-address-table MyRtgTable
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-address user regex
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# exit
Router(config-sbc-sbe-rtgpolicy)# rtg-src-domain-table MyRtgTable
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-domain cisco.com regex
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)# exit
Router(config-sbc-sbe-rtgpolicy)# exit
```

Example: Trunk-Group ID Routing

The following example shows how to configure the TGID routing to match the TGID parameters of a source or destination SIP URI.

```
Router# configure terminal
Router(config)# sbc mysbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-adj-sip)# tgid-routing
Router(config-sbc-sbe-adj-sip)# exit
Router(config-sbc-sbe)# call-policy-set 1
Router(config-sbc-sbe-rtgpolicy)# rtg-src-trunk-group-id-table MyRtgTable
Router(config-sbc-sbe-rtgpolicy-rtgtable)# entry 1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# action complete
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# dst-adjacency SIP-AS540-PSTN-GW2
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# match-type tgid
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# tgid-context example-domain tgid
trunkgroup1
Router(config-sbc-sbe-rtgpolicy-rtgtable-entry)# exit
Router(config-sbc-sbe-rtgpolicy-rtgtable)#
```