1. Introduction

This document provides configuration and provisioning guidance to enable SIP transparency on SBC Core systems. In addition to the configuration examples on the Session Border Controller (SBC), this document provides an introduction to key topics related to SIP headers and bodies on the SBC Core platform.

1.1 Audience

This document is intended for design engineers, system engineers and operations staff for the purpose of deploying SIP on a SBC Core system. Although this document provides some background on the concepts involved, the reader is expected to have a basic understanding of SIP.

1.2 Support

IMPORTANT

The Transparency Profile is the recommended method of configuring transparency on the SBC Core for new deployments as well as when applying additional transparency configurations to existing deployments. Do not use IP Signaling Profile flags in these scenarios because the flags will be retired in upcoming releases.

Refer to the SBC SIP Transparency Implementation Guide for additional information.

Info

The instructions, commands and references in this document apply to the SBC Core (SBC 5000 series, SBC 7000, SBC 5400, and SBC Software Edition Ribbon).

This document does not apply to SBC Edge (SBC 1000 and 2000 systems).
Ribbon Technical Support can be obtained using the following methods:

- Phone: +1 888-391-3434 (Toll-free) or +1 978-614-8589 (Direct)
- Web: https://support.sonus.net/display/PORTAL/Salesforce+Login

2. SIP Transparency

For some SIP elements, transparency is a frequently-debated topic. When transparency for a SIP header or body is desired, the user may often compare the element against a SIP Proxy which is a typical benchmark for significant transparency. Considered a popular comparison, this topic needs to be addressed up front when discussing SIP transparency.

2.1 SIP Proxy vs. SIP B2BUA

The SIP devices that connect most peers and endpoints are typically a SIP Proxy or Back-to-Back User Agent (B2BUA). The most transparent device is the SIP Proxy; its behaviors are primarily specified in RFC 3261 and are very basic in its message processing capabilities. The required transparency of a Proxy is one of its few strengths when compared to a B2BUA.

![Figure 1: SIP Transparency Spectrum](image)

Although an SBC is not defined in any IETF standard, it is most closely associated functionally with a SIP B2BUA (RFC 5853, 7092). Unless otherwise specified, this document will use B2UA and SBC terms interchangeably.

While RFC 3261 goes into detail describing the required behavior of a SIP Proxy, its description for a B2BUA could be considered somewhat terse: “Since it is a concatenation of a UAC [User Agent Client] and UAS [User Agent Server], no explicit definitions are needed for its behavior.” This statement notwithstanding, debate and research into the transparency behavior of a B2BUA continued, but seemingly without consensus. An often referenced IETF draft (draft-marjou-sipping-01) submitted to the SIPPING WG was not accepted as a working group document.

Admittedly, complete SIP transparency is not achievable due to the needs and requirements of changing some headers. Even a SIP Proxy is not completely transparent. In many scenarios the ability to control and even minimize transparency is a strength of a B2BUA/SBC. Some key selling points of an SBC highlight its ability to not be transparent:

- SIP Normalization (including arbitrary SIP Message Manipulation)
- Topology Hiding
- Protocol Translation
- Codec Transcoding (allowing a non-transparent SDP)

Fundamentally, the Ribbon SBC behaves as a SIP Back-to-Back User Agent (B2BUA) and not as a SIP Proxy. (If SIP Proxy behavior is actually needed then use of the Ribbon PSX Policy Server should be considered as it can be deployed specifically as a SIP Proxy or Redirector.) Unlike a standard SIP Proxy, the Ribbon SBC can provide a wide spectrum of SIP message transparency, from fairly transparent to almost completely non-transparent.

This document describes the SBC SIP transparency controls, how they behave, and how they interact. Some configuration examples using these transparency controls is also provided.

3. SBC SIP Transparency and Control Mechanisms

Since its inception, the SBC includes two related types of control flags: Relay Flags and Transparency Flags. Relay Flags primarily control SIP at the Request and Response level and are discussed later in a separate section. Transparency Flags control SIP headers and bodies that are generally not modified when received in a SIP message. While these controls are related, there is no direct overlap or precedence between them.

![Figure 2: SIP Message Flow](image)
3.1 Existing SBC Transparency Mechanisms

Prior to release 4.0, SIP header and body transparency was controlled primarily by the use of individual Transparency Flags, mostly within the IP Signaling Profile (IPSP; ipSignalingProfile > commonIpAttributes > transparencyFlags) and apply on the egress leg of a session (egress relative to the SIP message).

<table>
<thead>
<tr>
<th>Table 1: IP Signaling Profile Transparency Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>acceptContactHeader</td>
</tr>
<tr>
<td>acceptHeader</td>
</tr>
<tr>
<td>acceptLanguageHeader</td>
</tr>
<tr>
<td>alertInformationHeader</td>
</tr>
<tr>
<td>authcodeHeaders</td>
</tr>
<tr>
<td>callInfoHeader</td>
</tr>
<tr>
<td>contactHeader</td>
</tr>
<tr>
<td>errorInfo</td>
</tr>
<tr>
<td>externalBody</td>
</tr>
<tr>
<td>fromHeader</td>
</tr>
<tr>
<td>geolocation</td>
</tr>
<tr>
<td>geolocationError</td>
</tr>
<tr>
<td>geolocationRouting</td>
</tr>
<tr>
<td>historyInfo</td>
</tr>
<tr>
<td>maxForwardsHeader</td>
</tr>
<tr>
<td>mwiBody</td>
</tr>
</tbody>
</table>

Maximum of 144 unique unknown headers can be configured across all Header Transparency Profiles.
The message bodies are described in blue cells.

If a header or body did not have a specific flag on the SBC, it was treated as unknown, which meant it, along with any other SBC-unknown header, was controlled by the single unknownHeader flag (or unknownBody).

When a transparency flag was added for a header, it meant that the header was now known and that the unknownHeader flag no longer controlled it.

This methodology was problematic as headers transitioned from unknown to known on the SBC. It also meant that the unknownHeader flag was a very coarse control as it would allow any header that was unknown to the SBC.

The SBC introduced a more robust future-proofing mechanism in release 4.0 called the Transparency Profile. Ribbon SBC version 4.2 extends the Transparency Profile with similar support for SIP message bodies and the flexible ability to explicitly exclude some headers and/or methods.

### 3.2 SIP Transparency Profile

A Transparency Profile is a user-configurable profile allowing the user to transparently pass almost any SIP header/body through the SBC. It is no longer necessary for a user to request the SBC to create a specific Transparency Flag for the desired header/body.

Both already-known and previously-unknown SIP headers and bodies can be configured in a Transparency Profile. By default, no headers or message bodies are present in a Transparency Profile. If a received Content-Type header value matches any "Message Body" entry configured in the Transparency Profile, the SBC transparently passes the corresponding message body in the outgoing message. The SBC supports configuring up to 256 Transparency Profiles.

The following functionality is included:

- Ability to transparently pass "all" SIP headers and message body types.
- Selectively ignore transparency of select headers and message body types (useful when transparency is enabled for all headers and/or message body types).
- Exclude one or more methods for which transparency of headers or message body types is not needed.

When a header or body is specified in a Transparency Profile, the profile will take precedence over any applicable Transparency Flag. For headers not specified in a transparency profile, the setting of existing Transparency Flags will continue to determine the transparency of that header. In this way, a Transparency Profile can either override or augment existing Transparency Flag settings. This document will describe some usage scenarios where both mechanisms may be used together.

When configuring a Transparency Profile for specific SIP headers, Ribbon recommends that the unknownHeader flag be disabled (similarly, when configuring a Transparency Profile for specific SIP message bodies, Ribbon recommends that the unknownBody flag be disabled).

The following transparency is not supported by the SBC:

- ACK messages are not normally sent end-to-end through the SBC. Transparency of ACK messages is not supported even if the EndToEndAck flag is enabled in the "IP Signaling Profile".
- In late media scenarios, the SBC does not support transparency of headers and bodies.

For configuration details, see Service Profiles - Transparency Profile (EMA) or Transparency Profile - CLI.
3.2.1 SIP Message Header

The SBC introduced the Transparency Profile in release 4.0 to configure one or more SIP headers in a single profile to be passed transparently through. Release 4.2 extended the abilities of the Transparency Profile further. It now supported transparency for out-of-dialog messages, the ability to exclude specific headers from transparency and the ability to configure transparency on a per-method basis (e.g. INVITE, REGISTER, SUBSCRIBE, REFER, etc...), where specific methods can be excluded from transparency for that header. If no methods are specified to be excluded, then the configured header will be transparent for all methods.

```
set profiles services transparencyProfile <profile> sipHeader <SIP Header>
```

where <SIP Header> is case insensitive, supports up to 31 characters, and supports an "all" entry to match all headers (see section 3.3 for exceptions).

The ability to exclude specific headers from transparency is primarily intended for use in conjunction with the "all" header option.

SIP headers are also configurable using compact form. When configuring specific headers in a Transparency Profile, Ribbon recommends the configuration of both compact and long forms.

```
Refer to IANA for the SIP header fields and their compact forms at: http://www.iana.org/assignments/sip-parameters/sip-parameters.xhtml
```

Compact form can be received by the SBC, but the Ribbon SBC never generates the Compact form of any headers.

The Ribbon SBC does not send multiple header instances as a comma separated list; they are always sent as separate headers.

The following SIP headers are not controlled by the Transparency Profile (or any Transparency Flags), and are ignored if configured in a Transparency Profile:

- Allow
- Call-ID
- CSeq
- Max-Forwards
- Require
- RSeq
- Supported
- RAck
- P-Associated-URI

**Note**

The transparency of Allow, Supported, and Require headers can be controlled by using SIP Param Filter Profile. For more information, refer to SIP Param Filter Profile.

If Contact Header is specified in a Transparency Profile, then it is treated as full Contact transparency and it will take precedence over other Contact related flags (such as useZoneLevelDomainNameInContact).

3.2.2 SIP Message Body

Ribbon SBC version 4.2 extends the Transparency Profile with similar support for SIP message bodies. In addition, both message header and body transparency is configurable on a per-method basis (e.g. INVITE, REGISTER, SUBSCRIBE, REFER, etc...), where specific methods can be excluded from transparency for that body. If no methods are specified to be excluded, then the configured body is transparent for all methods.

```
set profiles services transparencyProfile <profile> sipMessageBody <Content-Type>
```
where `<Content-Type>` is case insensitive, supports up to 127 characters, and supports an "all" entry to match all message bodies except those described in the below list.

The following Content-Types are not controlled by the Transparency Profile and are ignored if configured in a profile:

- application/sdp
- application/dtmf
- application/dtmf-relay
- application/sonus-media
- application/broadsoft
- application/isup
- multipart/mixed
- multipart/alternative

Multipart/mixed and multipart/alternative are ignored because the SBC automatically matches each component message body contained within a multipart message independently. For example, if "application/qsig" is configured in a profile, the SBC will match it even if it is contained within a multipart/mixed message with no additional configuration needed.

A Transparency Profile cannot control the SDP (application/sdp). The SDP and its controls will be discussed later in this document.

The other exceptions are due to existing Relay Flags (see table below) elsewhere within the SBC.

<table>
<thead>
<tr>
<th>Relay Flag</th>
<th>Configuration Location</th>
<th>Content Applicability</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtmfBody</td>
<td>IP Signaling Profile</td>
<td>application/dtmf and application/dtmf-relay</td>
</tr>
<tr>
<td>sonusMediaBody</td>
<td>IP Signaling Profile</td>
<td>application/sonus-media</td>
</tr>
<tr>
<td>thirdPartyBodies</td>
<td>IP Signaling Profile</td>
<td>application/broadsoft</td>
</tr>
<tr>
<td>isupMimeBodyRelay</td>
<td>SIP Trunk Group</td>
<td>application/isup</td>
</tr>
<tr>
<td>conferenceEventPackage</td>
<td>IP Signaling Profile</td>
<td>application/conference-info+xml</td>
</tr>
<tr>
<td>dialogEventPackage</td>
<td>IP Signaling Profile</td>
<td>application/dialog-info+xml</td>
</tr>
</tbody>
</table>

See Relay Flags below for details.

### 3.2.3 Inter-working with IP Signaling Profile

SIP Transparency Profile provides advanced control of the transparency of headers and message bodies. However, customers may continue with the existing (albeit simple) IPSP transparency controls in PSX/e-PSX/ERE.

Using message body transparency as an example:

- If a message body is allowed by the IPSP but configured to be ignored by the Transparency Profile, it is not transparently passed through.
- If a message body is allowed by the IPSP but configured to be excluded by the Transparency Profile for a particular SIP method, it is not transparently passed through for that specific SIP method.
- If specific message bodies are allowed by the IPSP, but transparency of ‘all’ message bodies is configured in the Transparency Profile, all types of message bodies are transparently passed through.
- If ‘Unknown Body’ transparency is enabled in the IPSP, but an unknown message body is configured to be ignored (or excluded for a specific SIP method) by the Transparency Profile, it is not transparently passed through.

### 3.3 SIP Header Transparency

Header transparency is based on the headers that are present in the Transparency Profile of the egress trunk group for requests and headers that are present in the ingress trunk group for responses. By default, no headers are present in the Transparency Profile.
3.3.1 Allowed Values for a SIP Header in the Transparency Profile

A 'sipHeader' in the Transparency Profile can be composed of:

- Any string with a maximum length of 31 characters
- Any case, lower/upper/mixed
- Special characters are allowed

3.3.2 SIP Headers not Under Transparency Control in Relay Scenarios

Some headers are not under the control of transparency flags in relay scenarios. These headers can be classified into three categories as shown in the below table:

**Table 3**: SIP Headers not Under Control of Transparency Flags in Relay Scenarios

<table>
<thead>
<tr>
<th>Header Classifications</th>
<th>Added/modified by the SBC</th>
<th>Not sent at all</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transiently sent irrespective of settings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accept-Language</td>
<td>Also</td>
<td>Allow</td>
</tr>
<tr>
<td>Alert-Info</td>
<td>Anonymity</td>
<td>Max-Forwards</td>
</tr>
<tr>
<td>Authorization</td>
<td>Diversion</td>
<td>Require</td>
</tr>
<tr>
<td>Content-Length</td>
<td>Path</td>
<td>Supported</td>
</tr>
<tr>
<td>Error-Info</td>
<td>P-Charge-Info</td>
<td>RAck</td>
</tr>
<tr>
<td>Event</td>
<td>P-DCS-Billing-Info</td>
<td>P-Associated-URI</td>
</tr>
<tr>
<td>Expires</td>
<td>P-K-Cfl</td>
<td></td>
</tr>
<tr>
<td>Min-Expires</td>
<td>P-K-Cfo</td>
<td></td>
</tr>
<tr>
<td>Min-SE</td>
<td>P-Preferred-Identity</td>
<td></td>
</tr>
<tr>
<td>Proxy-Authenticate</td>
<td>P-Sig-Info</td>
<td></td>
</tr>
<tr>
<td>Proxy-Authorization</td>
<td>Record-Route</td>
<td></td>
</tr>
<tr>
<td>Proxy-Require</td>
<td>Remote-Party-ID</td>
<td></td>
</tr>
<tr>
<td>RACK</td>
<td>Reply-To</td>
<td></td>
</tr>
<tr>
<td>Reason</td>
<td>Requested-By</td>
<td></td>
</tr>
<tr>
<td>Refer-Sub</td>
<td>Service-Route</td>
<td></td>
</tr>
<tr>
<td>Resource-Priority</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retry-After</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSeq</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Session-Expires</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Headers may be configured in compact form and transparently passed using a Transparency Profile. It is advisable to configure both compact and long formats to ensure both types of received headers in the PDU are transparently passed.
3.3.3 SIP Headers Not Transparently Passed in Calls

The following SIP headers are not supported by a Transparency Profile (or any Transparency flags):

- Call-ID
- RSeq
- Allow
- CSeq
- Max-Forwards
- Require
- Supported
- RAck
- P-Associated-URI

These SIP headers are entirely added and/or modified by the SBC itself and cannot be transparently passed.

3.3.4 SIP Headers Brought Under Transparency Control in Relay Scenarios

Previously, the following headers were transparently passed by the SBC. From release 4.2 onwards, these headers are controlled using transparency flags.

- Accept
- SIP-ETag
- SIP-If-Match
- Suppress-If-Match

3.3.5 SIP Header Transparency Behaviors

SBC transparency mechanisms control the initial INVITE, its responses, and other requests/responses within the INVITE dialog, as well as REGISTER, BYE, UPDATE, REFER, INFO, PUBLISH, MESSAGE, OPTIONS, SUBSCRIBE, NOTIFY requests and their responses (this assumes that the request method has been allowed by the applicable Relay Flag: INFO, MESSAGE, etc…).

There are some exceptions to the transparency mechanisms. Some SIP Methods and some SIP headers are not affected by any configurable transparency mechanism, while other headers may not be affected by transparency controls in some scenarios (in-dialog vs. out-of-dialog).

3.3.6 Non-Transparent Methods and Scenarios

The following SIP method is not supported by a Transparency Profile (or any Transparency flags):

- ACK (even if the endToEndAck flag is enabled)

3.3.7 Transparently Pass or Block Option Tags/Methods of Allow/Require/Supported Headers

Option tags/methods of the following SIP headers can be transparently passed or blocked by configuring the SIP Param Filter Profile.

- Allow
- Require
- Supported

Note

For configuration details, see:
3.3.8 In-Dialog vs. Out-of-Dialog

Some header behaviors vary depending on whether they are received in or out of an existing dialog. While the Transparency Profile has been extended in 4.2 to apply to out-of-dialog messages, there are some specific headers whose behavior is not under the control of a Transparency Profile (or Transparency Flags) when received in out-of-dialog messages.

A Dialog "represents a peer-to-peer SIP relationship between two user agents that persists for some time. The dialog facilitates sequencing of messages between the user agents and proper routing of requests between both of them. The dialog represents a context in which to interpret SIP messages." (reference: RFC 3261)

The SBC can receive messages within a dialog or outside of a dialog, and treats them differently based upon that relationship (or lack thereof).

If "Session-Expires" header is configured, the SBC passes the "Min-SE" header and their values transparently to the egress remote peer. This behavior is achieved either by configuring "all" header or "Session-Expires" and "Min-SE" header in the Transparency Profile. The Keep-Alive behavior must be disabled by setting "Session-Expires" to "0" in the respective SIP Trunk Group.

For example,

```plaintext
set profiles services transparencyProfile ALL_HEADER sipHeader all
set profiles services transparencyProfile SIP_HEADER sipHeader Session-Expires ignoreTransparency no
set profiles services transparencyProfile SIP_HEADER sipHeader Min-SE ignoreTransparency no
set addressContext default zone Zone1 sipTrunkGroup TG1 signaling timers sessionKeepalive 0
```

If "all" header in Transparency Profile is configured and SBC supports "Session Keep Alive" mechanism, the "Session-Expiry" and "Min-SE" headers are excluded from the Transparency Profile by setting `ignoreTransparency` as "yes".

For example,

```plaintext
set profiles services transparencyProfile ALL_HEADER sipHeader all
set profiles services transparencyProfile ALL_HEADER sipHeader Session-Expires ignoreTransparency yes
set profiles services transparencyProfile ALL_HEADER sipHeader Min-SE ignoreTransparency yes
```

Out-of-Dialog header behavior irrespective of the Transparency Profile or Flags:

**Table 4: Out-of-Dialog SIP Header Behavior**

<table>
<thead>
<tr>
<th>SIP Header</th>
<th>Out-of-Dialog Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept-Language</td>
<td>Sent</td>
</tr>
<tr>
<td>Alert-Info</td>
<td>Sent</td>
</tr>
<tr>
<td>Also</td>
<td>Dropped</td>
</tr>
<tr>
<td>Anonymity</td>
<td>Dropped</td>
</tr>
<tr>
<td>Authorization</td>
<td>Sent</td>
</tr>
<tr>
<td>Content-Length</td>
<td>Sent</td>
</tr>
<tr>
<td>Diversion</td>
<td>Dropped</td>
</tr>
<tr>
<td>Error-Info</td>
<td>Sent</td>
</tr>
<tr>
<td>Event</td>
<td>Sent</td>
</tr>
<tr>
<td>Field</td>
<td>Status</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Expires</td>
<td>Sent</td>
</tr>
<tr>
<td>Min-Expires</td>
<td>Sent</td>
</tr>
<tr>
<td>Min-SE</td>
<td>Sent</td>
</tr>
<tr>
<td>P-Charge-Info</td>
<td>Dropped</td>
</tr>
<tr>
<td>P-DCS-Billing-Info</td>
<td>Dropped</td>
</tr>
<tr>
<td>P-K-Cfl</td>
<td>Dropped</td>
</tr>
<tr>
<td>P-K-Cfo</td>
<td>Dropped</td>
</tr>
<tr>
<td>P-Preferred-Identity</td>
<td>Dropped</td>
</tr>
<tr>
<td>P-Sig-Info</td>
<td>Dropped</td>
</tr>
<tr>
<td>Path</td>
<td>Dropped</td>
</tr>
<tr>
<td>Proxy-Authenticate</td>
<td>Sent</td>
</tr>
<tr>
<td>Proxy-Authorization</td>
<td>Sent</td>
</tr>
<tr>
<td>Proxy-Require</td>
<td>Sent</td>
</tr>
<tr>
<td>RAck</td>
<td>Sent</td>
</tr>
<tr>
<td>Reason</td>
<td>Sent</td>
</tr>
<tr>
<td>Record-Route</td>
<td>Dropped</td>
</tr>
<tr>
<td>Refer-Sub</td>
<td>Sent</td>
</tr>
<tr>
<td>Remote-Party-ID</td>
<td>Dropped</td>
</tr>
<tr>
<td>Reply-To</td>
<td>Dropped</td>
</tr>
<tr>
<td>Requested-By</td>
<td>Dropped</td>
</tr>
<tr>
<td>Resource-Priority</td>
<td>Sent</td>
</tr>
<tr>
<td>Retry-After</td>
<td>Sent</td>
</tr>
<tr>
<td>RSeq</td>
<td>Sent</td>
</tr>
<tr>
<td>Service-Route</td>
<td>Dropped</td>
</tr>
<tr>
<td>Session-Expires</td>
<td>Sent</td>
</tr>
<tr>
<td>Subscription-State</td>
<td>Sent</td>
</tr>
<tr>
<td>Warning</td>
<td>Sent</td>
</tr>
<tr>
<td>WWW-Authenticate</td>
<td>Sent</td>
</tr>
</tbody>
</table>

### 3.4 SIP Message Body Transparency

Message body transparency is based on message bodies that are present in the Transparency Profile of the egress trunk group for requests and content-types/bodies that are present in the ingress trunk group for responses. By default, no message bodies are present in the Transparency Profile.
3.4.1 Allowed Values for a Content Type in the Transparency Profile

The allowed range for a "contentType" in the Transparency Profile includes:

- Any string with a maximum length of 127 characters
- Any case, lower/upper/mixed
- Special characters are allowed
- An existing transparency flag for that Content-Type in IP Signaling Profile (IPSP) is not required

3.4.2 Transparency of Multi-part Message Bodies

The SBC treats constituent parts of a 'multipart/mixed' message body just as it treats any message with a single body. Therefore, a constituent part of a 'multipart/mixed' message body will be transparently passed through only if the Content-Type specified in the MIME envelope of the corresponding part has been configured in the Transparency Profile.

For example, consider a SIP message with content-type 'multipart/mixed' with two parts in its body: the first part is type 'application/foo' and the other type 'application/bar'. If the Transparency Profile is configured to transparently pass 'application/foo', then the first part of the message body is passed transparently in the egress SIP message.

3.4.3 Support for Message Body Transparency Across Ribbon Gateways

This feature will be supported across Ribbon Gateways (using Ribbon Proprietary GW to GW Signalling) only for SIP INVITE messages.

3.4.4 SDP

The SBC supports anchoring the following media types:

1. Audio
2. Video Main
3. Video Extended (for Content Share)
4. Binary Floor Control Protocol (UDP and TCP)
5. Far End Camera Control (FECC)
6. Message Session Relay Protocol (MSRP)

For all the above mentioned media types (with the exception of Audio), the SBC consumes (hence does not transparently relay) the following attributes that are required to anchor the media:

- C line
- RTCP attributes
- Media direction (a=sendrecv/sendonly/inactive/recvonly)

For Audio, the SBC does not transparently relay theptime and maxptime attributes in addition to the above mentioned attributes.

You must enable Video (assign a valid video bandwidth) and Audio transparency to achieve the above described behavior using the below CLI syntax.

**Note**

Associate the following configuration with both Trunk Groups.
3.4.4.1 SDP Transparency Flag

Make note that the `sdpTransparencyState` signaling object within the SIP Trunk Group must not be considered a general use parameter. It is specific to some functionality (mainly ICE) and environments; however, this flag does not apply to all types of call flows.

**Note**

Do not enable the `sdpTransparencyState` flag unless specifically directed to do so by Ribbon Design or Support engineers.

3.4.5 Audio Transparency

The SBC for the audio m line allows relaying unknown attributes. The SBC allows transparency for subset of the attributes like `rtpmap`, `fmtp`, and `T38 fax`. Audio transparency functionality is used to manage bandwidth for audio stream in the pass-through calls. By enabling this feature, audio codecs that are unknown to the system are available to establish audio calls or streams.

SBC supports audio transparency for known attributes by relaying attributes and codecs transparently in pass-through scenarios for SIP-SIP calls only. However, the following exceptions require system handling:

- `recvonly/sendonly/sendrecv/inactive`
- `crypto`
- `X-dmi`
- `rtcp`
- `fingerprint`
- `OMR`

**Note**

This feature does not support H323-H323 and GW-GW calls.

Audio Transparency Feature is controlled by two flags:

- Enable Transcoder-Free-Transparency for the session (enable on either of the PSPs).
- Enable Selective-SDP-Transparency on both ingress and egress Trunk Groups that receive the relayed SDP.

Bandwidth (b) lines are transparently relayed and do not play any role in calculating the unknown audio codec bandwidth. The following PSP configuration bits for Audio Transparency feature are included for Unknown audio bandwidth reservation to calculate the Unknown audio bandwidth:

- `unknownCodecBitRate`
- `unknownCodecPacketSize`

**Note**

If the bandwidth is not configured, the default settings (Packet Size—10 ms and Bit Rate—124 KB/s) are used for a pass-through call.

3.4.5.1 Audio Transparency and Reserve Bandwidth for Preferred Common Codec

By default for pass-through calls, SBC reserves the worst case common audio codec bandwidth on Trunk Groups and IP interfaces, and polices for the same bandwidth. To facilitate pass-through calls scenarios/cases, where media uses the preferred common codec the flag `reserveBwFo`
rPreferredAudioCommonCodec is added to reserve the bandwidth associated with the preferred common codec (instead of the worst case common codec) on the Trunk Groups and IP interfaces. When this flag is enabled, bandwidth of the first common codec from Answer (SIP) is used for reservation and bandwidth of the heaviest common codec is used for policer.

**Note**
This flag can be used independently or in conjunction with Audio Transparency feature and/or policeOnHeaviestAudioCodec flag. This functionality is currently supported for SIP-SIP call scenarios only. In the event that policeOnHeaviestAudioCodec and reserveBwForPreferredAudioCommonCodec are both configured, the following behavior applies:

- `reserveBwForPreferredAudioCommonCodec` impacts the bandwidth reservation policy. That is, first common codec from Answer (SIP) and,
- `policeOnHeaviestAudioCodec` impacts the policer configuration. That is, heaviest codec in the offer or answer.

**Note**
The flag `reserveBwForPreferredAudioCommonCodec` is active for a call when both the PSPs have this flag enabled. If this flag is disabled in any of the PSPs, the flag is not applied.

### 3.4.5.2 Media Policer Reservation For Worst Case Codec

By default for pass-through calls, the SBC reserves the worst case common audio codec bandwidth on trunk groups and IP interfaces, and polices for the same bandwidth. To facilitate asymmetric pass-through calls scenarios/cases and to police on the heaviest codec in the offer or answer, the flag `policeOnHeaviestAudioCodec` is used in the PSP.

**Note**
This flag can be used independent or in conjunction with Audio transparency feature and/or `reserveBwForPreferredAudioCommonCodec` flag. This functionality is currently supported for SIP-SIP call scenarios only.

### 3.4.5.3 Configuring Audio Transparency

Configuring the basic audio transparency feature contains:

- **Enabling the sdpAttributesSelectiveRelay Parameter on Both Ingress and Egress Trunk Groups**
- **Configuring the transcoderFreeTransparency Parameter on Packet Service Profile**
- **Configuring audioTransparecy Parameter on Packet Service Profile**

#### Enabling the sdpAttributesSelectiveRelay Parameter on Both Ingress and Egress Trunk Groups

```bash
set addressContext default zone ZONE1 sipTrunkGroup TG_SBX_INT media sdpAttributesSelectiveRelay enabled
set addressContext default zone ZONE2 sipTrunkGroup TG_SBX_EXT media sdpAttributesSelectiveRelay enabled
```

#### Configuring the transcoderFreeTransparency Parameter on Packet Service Profile

```bash
set profiles media packetServiceProfile PSP_INT packetToPacketControl transcode transcoderFreeTransparency
set profiles media packetServiceProfile PSP_EXT packetToPacketControl transcode transcoderFreeTransparency
```
3.5 Transparency Profile Usage

As discussed previously, the Transparency Profile does not deprecate any existing Transparency Flag. Those flags continue to function as designed. When a header/body is specified in a Transparency Profile, then the profile takes precedence over any applicable Transparency Flag. For headers/bodies not specified in a transparency profile, the setting of existing Transparency Flags continues to determine the transparency of that header.

When configuring a Transparency Profile for specific SIP headers, SBC recommends disabling the unknownHeader flag (similarly, when configuring a Transparency Profile for specific SIP message bodies, SBC recommends disabling the unknownBody flag).

There are three modes of using Transparency Profile:

- Transparency Profile with Specific Headers (Positive Enumeration)
- Maximum Transparency Using the Transparency Profile
- Transparency Profile, All Headers with Exceptions (Negative Enumeration)

3.5.1 Transparency Profile with Specific Headers (Positive Enumeration)

In this mode, only headers/bodies explicitly configured in the Transparency Profile are allowed to pass-through.

For example, the following scenario allows only the headers "p-asserted-identity" and "xyzHdr" and message bodies of type "application/simple-message-summary" and "xyzContentType" to pass transparently.

```plaintext
set profiles services transparencyProfile ALLOW_SPECIFIC_HDRS_BODIES sipHeader p-asserted-identity
set profiles services transparencyProfile ALLOW_SPECIFIC_HDRS_BODIES sipHeader xyzHdr
set profiles services transparencyProfile ALLOW_SPECIFIC_HDRS_BODIES sipMessageBody application/simple-message-summary
set profiles services transparencyProfile ALLOW_SPECIFIC_HDRS_BODIES sipMessageBody xyzContentType
set profiles services transparencyProfile ALLOW_SPECIFIC_HDRS_BODIES state enabled
commit
```

3.5.2 Maximum Transparency using the Transparency Profile

Complete or maximum transparency is occasionally desired, especially during initial integration testing to determine if specific headers are required for the success of certain call flows.

```plaintext
set profiles services transparencyProfile MAX_TRANSPARENCY sipHeader all
set profiles services transparencyProfile MAX_TRANSPARENCY sipMessageBody all
set profiles services transparencyProfile MAX_TRANSPARENCY state enabled
commit
```

Note

For configuring Bit Rate (kbps), Packet Size (ms) and Reserve BW For Preferred Audio Common Codec for pass-through calls flags on PSX, refer to PSX Documentation.
Additional Relay Flags also need to be enabled to maximize the transparency of the Trunk Group for testing. See Relay Flags above.

3.5.3 Transparency Profile, All Headers with Exceptions (Negative Enumeration)

All headers/bodies are allowed to pass-through unless they are explicitly disallowed by Transparency Profile configuration. The ignoreTransparency header option within the Transparency Profile is primarily used for excluding one or more specific headers when paired with the "all" header option. In the example below, the user wishes to pass all SIP headers except for the History-Info header.

For example, in the following scenario, the rules are configured and all the headers and message bodies except "history-info" and "application/resource-lists+xml" are passed transparently. The "xyzHeader" is passed transparently in all methods except INFO and REGISTER. The "xyzContentType" is passed transparently in all methods except INVITE.

```plaintext
set profiles services transparencyProfile ALMOST_ALL_TRANSPARENCY sipHeader all
set profiles services transparencyProfile ALMOST_ALL_TRANSPARENCY sipMessageBody all
set profiles services transparencyProfile ALMOST_ALL_HDRS sipHeader history-info ignoreTransparency yes
set profiles services transparencyProfile ALMOST_ALL_HDRS sipMessageBody application/resource-lists+xml ignoreTransparency yes
set profiles services transparencyProfile ALMOST_ALL_HDRS sipHeader xyzHeader excludedMethods info,register
set profiles services transparencyProfile ALMOST_ALL_HDRS sipMessageBody xyzContentType excludedMethods invite
set profiles services transparencyProfile ALMOST_ALL_HDRS state enabled
commit
```

The excludedMethods parameter indicates the list of methods for which the transparency is not allowed and is common to both header and body entries.

```plaintext
set profiles services transparencyProfile TP1 sipHeader all
set profiles services transparencyProfile TP1 sipHeader all excludedMethods bye,info,notify
commit
```

**Note**

If a specific header is configured, ignoreTransparency and excludedMethods are exclusive. Once ignoreTransparency is configured, excludedMethods no longer displays. Likewise, once excludedMethods is configured, ignoreTransparency no longer displays.

**Note**

Use the ignoreTransparency flag in a configuration to allow all SIP headers, or use it when one of the SIP header transparency flags is enabled in the IP Signaling Profile.

3.5.4 Existing Deployment Augmented with a Transparency Profile

Existing deployments will likely utilize Transparency Flags, and those that must pass proprietary or otherwise SBC unsupported SIP headers will most likely make use of the unknownHeader transparency flag in an IP Signaling Profile.

While a Transparency Profile can be configured to completely overlap with any existing Transparency Flags settings, it is not required. A Transparency Profile can be configured to simply augment existing Transparency Flags settings with a more surgical configuration and allowing unknownHeader to be disabled.

For example, a user may wish to have the SBC transparently pass RFC 4474 identity headers. Prior to the introduction of the Transparency Profile, the user would have had to enable the unknownHeader transparency flag.

Rather than continue to allow all unknown headers through the SBC, the user can configure a Transparency Profile that only allows the RFC 4474 identity headers (configured in standard and compact forms) and disable the unknownHeader transparency flag.
3.6 SRTP Pass-through

SBC supports SRTP media pass-through for SRTP and SRTCP media streams. SBC does not terminate the SDP security description or SRTP media streams and passes them through without authenticating, decrypting, and encrypting. In this pass-through mode of operation, SBC treats SRTP media as plain text RTP pass-through media.

The following diagram illustrates the media flow for an SRTP pass-through call.

**Figure 3: SRTP Packet to Packet Media Call Flow**

To control this SRTP media pass-through, the `allowPassthru` flag is available from the `secureRtpRtcp` parameter of the PSP. When `allowPassthru` flag is enabled along with the security `enableSrtp` flag, it allows SBC to pass-through SRTP media without authenticating, decrypting, and encrypting it internally. When selected, this flag prioritizes SRTP pass-through media over terminated SRTP media. When disabled, this flag terminates all SRTP and SRTCP media for authentication, encryption, or decryption. This flag is disabled by default.

3.7 Relay Flags

Relay Flags exist mostly within the IP Signaling Profile (IPSP: `ipSignalingProfile > commonIpAttributes > relayFlags`) and apply on the ingress leg of a session (ingress relative to the SIP message).

Relay Flags are intended mainly for SIP Methods (Requests) and Responses (and some SIP message bodies) that normally get consumed or modified by the SBC when received in the incoming SIP message.

Albeit imprecise, a good method to contrast Relay Flags and Transparency Flags/Profiles is to consider that *Relay* controls whether a SIP request/response is sent through the SBC, while the *Transparency* controls whether a header/body in a SIP request/response is sent through the SBC.

**Table 5: Configuration Locations of Relay Flags**

<table>
<thead>
<tr>
<th>Relay Flags</th>
<th>Configuration Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>conferenceEventPackage</td>
<td>IP Signaling Profile &gt; Common IP Attributes</td>
</tr>
</tbody>
</table>
### 3.8 Relaying REFER Request

The SBC is enhanced to relay REFER request, even though the refer relay flag is disabled. To support this enhancement, Conditional Relay Matching criteria is provided by the SBC. Using this criteria, SBC decides whether to relay and process the REFER request message or not.

If the refer relay flag is disabled, the Call Control (CC) mechanism forwards the REFER request to Digital Signaling (DS). DS exchanges information with the PSX to check the match criteria set in Conditional Relay Matching.

The matched criteria includes call parameters such as Username, Directory Number (DN), or Fully Qualified Domain Name (FQDN).

- If the call parameters received with the REFER request match the call routing criteria, SBC relays the REFER request to Egress SIPSG.
- If the call parameters received with the REFER request do not match the call routing criteria, the REFER request is processed locally by SBC. The REFER request acts as the transferor and the call is forwarded to the Egress SIPSG, resulting in call bridging. In this scenario, SBC sends back a 202 response and proceeds for local processing.

**Note**
If a REFER request is sent after a switchover and:

- If the refer relay flag is enabled, SBC relays REFER request.
- If the refer relay flag is disabled and DN/username/FQDN match, SBC relays the REFER request.
- If the refer relay flag is disabled and no DN/username/FQDN match, the REFER request is rejected. SBC cannot locally process the REFER request.
This feature is supported only for Blind/Unattended Transfer calls and not for Attended Transfer (refer with replaces) calls.

Figure 4: The following figure shows the enhanced REFER request processing call flow:

3.8.1 Configuring SBC For Enhanced Refer Processing

To configure this feature, perform the following steps:

1. Configure SBC for regular REFER call Blind Transfer.

2. Create SIP_MSG_TYPE_REFER call parameter filter profile (CPFP) in the PSX. Execute the following command to view the CPFP SIP_MSG_TYPE_REFER. This profile is already present in ERE.

   Info
   For more information on creating CPFP, refer to PSX Documentation.
> show table profiles callParameterFilterProfile
Description:
Profile used for routing based on SIP message type.

Possible completions:
- SIP_MSG_TYPE_INFO - SIP Message Type is Info
- SIP_MSG_TYPE_MESSAGE - SIP Message Type is Message
- SIP_MSG_TYPE_NOTIFY - SIP Message Type is Notify
- SIP_MSG_TYPEREFER - SIP Message Type is Refer
- SIP_MSG_TYPE_REGISTER - SIP Message Type is Register
- SIP_MSG_TYPE_SUBSCRIBE - SIP Message Type is Subscribe
- none - seed data for provisioning support

All > Profiles > Call Parameter Filter Profile

Figure 5: Call Parameter Filter Profile List showing SIP_MSG_TYPE_REFER profile

![Call Parameter Filter Profile List showing SIP_MSG_TYPE_REFER profile](image)

**Note**
A new script SONS_SIP_REFER_RELAY is seeded in both ERE and PSX.

3. Disable the Refer relay flag in IPSP.

```plaintext
set profiles signaling ipSignalingProfile DEFAULT_SIP commonIpAttributes relayFlags refer disable
```

4. Enable the Notify relay in Egress side on IPSP to relay REFER for DN/Username/FQDN match.

```plaintext
set profiles signaling ipSignalingProfile DEFAULT_SIP commonIpAttributes relayFlags notify enable
```

All > Profiles > Signaling > Ip Signaling Profile > Common Ip Attributes > Relay Flags

Figure 6: Notify and Refer relay flags
5. Create a new routing label with the script SONS_SIP_REFER_RELAY to trigger process refer request feature.

![Routing Label Screen](image)

**Note**

The routing label action must be set as script.

```
set global callRouting routingLabel <routing_label> script SONS_SIP_REFER_RELAY action script
```

**Figure 7**: Creating a Routing Label

**Figure 8**: Routing Label screen
5. Configure a DN criteria in the standard route and attach the SIP_MSG_TYPEREFER profile to the standard route by executing the following command:

```
set global callRouting route none Sonus_NULL Sonus_NULL standard <Matched_DN or FQDN> 1 all ALL SIP_MSG_TYPEREFER Sonus_NULL routingLabel <routing_label>
```

For DN (Directory Number) or username

```
set global callRouting route none Sonus_NULL Sonus_NULL standard <Matched_DN or Username> 1 all ALL SIP_MSG_TYPEREFER Sonus_NULL routingLabel <routing_label>
```

For FQDN with DN or username

```
set global sipDomain <Matched_domain_name>
set global callRouting route none Sonus_NULL Sonus_NULL standard <Matched_DN or Username> 1 all ALL SIP_MSG_TYPEREFER <Matched_domain_name> routingLabel <routing_label>
```

All > Global > Call Routing > Route

**Figure 9:** Creating a Route
Figure 10: Route screen

7. Execute the following commands to view the call detail status and call media status.

```bash
show status global callDetailStatus
or
show status global callMediaStatus
```

3.9 SIP Param Filter Profile

The SBC is enhanced to support SIP Param Filter Profile to allow the operator to create a profile defining a set of SIP header tags and methods to transparently pass or block, and then assign that profile to a trunk group. The SIP headers configured in this profile for pass-through are transparently passed to the Egress trunk group if received in the Ingress SIP message.

The SIP Param Filter Profile includes the following characteristics:

- This profile takes precedence over existing mechanism/flags when transparently passing Allow/Supported/Require headers, but does not
impact corresponding configurations established by the operator. It is operators responsibility to ensure the system is configured properly
so that transparently-passed values do not conflict with existing configurations. For example, do not configure 100rel as pass-through if
100rel support fro SIP Trunk Group Signaling is disabled.

- The settings of SIP Param Filter Profiles for both ingress and egress legs dictate the actual pass-through results (see SIP Param Filter
Profile Behavior table below for details.)
- Pass-through of individual header values is configurable.
- SIP tags are provided for unknown SIP parameter transparency only. Known SIP parameter transparency is still determined using
existing SBC application logic (from Ingress leg to Egress leg) and configurations.

![Note]
The SBC supports configuring up to 32 SIP Param Filter Profiles. Each profile can be configured using any/all of the SIP headers Allow/Supported/Require.

The SIP Param Filter Profile Behavior table explains the SIP Param Filter Profile behavior when using the Allow, Supported and Require headers.

**Table 6: SIP Param Filter Profile Behavior**

<table>
<thead>
<tr>
<th>Header</th>
<th>SIP Param Filter Profile Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow</td>
<td><strong>Ingress leg:</strong></td>
</tr>
<tr>
<td></td>
<td>The Allow header in the received message is processed after Ingress SIP Message Manipulation (SMM) processing but before any other SBC processing occurs.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Pass-through &lt;method list&gt;</strong> – Any methods present in the Allow header but not included in the method list are removed.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Pass-through &quot;all&quot;</strong> – All methods present in the Allow header are left intact.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Block &lt;method list&gt;</strong> – Methods specified in the &lt;method list&gt; which exist in the Allow header are removed.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Block &quot;all&quot;</strong> – All methods present in the Allow header are removed.</td>
</tr>
<tr>
<td></td>
<td><strong>Egress leg:</strong></td>
</tr>
<tr>
<td></td>
<td>The Allow header in the message to be egressed by SBC is processed after all SBC processing but before Egress SMM processing is performed.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Pass-through &lt;method list&gt;</strong> – Any methods present in the Allow header to be egressed but not specified in the &lt;method list&gt; are removed.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Pass-through &quot;all&quot;</strong> – All methods present in the Allow header are left intact.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Block &lt;method list&gt;</strong> – Any methods present in the &lt;method list&gt; are removed if they exist in the Allow header.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Block &quot;all&quot;</strong> – All methods present in the Allow header are removed.</td>
</tr>
<tr>
<td>Require/Supported</td>
<td><strong>Ingress leg:</strong></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td><strong>Pass-through <code>&lt;option tag list&gt;</code></strong> – Any Option tags present in the Require/Supported header but not specified in the <code>&lt;option tag list&gt;</code> are removed.</td>
</tr>
<tr>
<td></td>
<td><strong>Pass-through &quot;all&quot;</strong> – All Option tags present in the Require/Supported header are left intact.</td>
</tr>
<tr>
<td></td>
<td><strong>Block <code>&lt;option tag list&gt;</code></strong> – Any Option tags present in the <code>&lt;option tag list&gt;</code> are removed if they exist in the Require/Supported header.</td>
</tr>
<tr>
<td></td>
<td><strong>Block &quot;all&quot;</strong> – All Option tags present in the Require/Supported header are removed.</td>
</tr>
<tr>
<td><strong>Egress leg:</strong></td>
<td>The Require/Supported header in the message to be egressed by SBC is processed after all SBC processing but before Egress SMM processing occurs.</td>
</tr>
<tr>
<td></td>
<td><strong>Pass-through <code>&lt;option tag list&gt;</code></strong> – Any Option tags present in the Require/Supported header to be egressed but not in the <code>&lt;option tag list&gt;</code> are removed.</td>
</tr>
<tr>
<td></td>
<td><strong>Pass-through &quot;all&quot;</strong> – All Option tags present in the Require/Supported header are left intact.</td>
</tr>
<tr>
<td></td>
<td><strong>Block <code>&lt;option tag list&gt;</code></strong> – Any Option tags present in the <code>&lt;option tag list&gt;</code> are removed if they exist in the Require/Supported header.</td>
</tr>
<tr>
<td></td>
<td><strong>Block &quot;all&quot;</strong> – All Option tags present in the Require/Supported header are removed.</td>
</tr>
</tbody>
</table>

If an option tag present in the received Ingress request is dropped due to Require transparency settings, and if `rejectRequest` is configured, the request is rejected with a new internal cause code. This internal cause code is mapped to 420 "Bad Extension" by default. Option tags added to Require header due to SBC processing (e.g. path) are not rejected even if they are eventually dropped by the Require transparency functionality.

### 3.10 SIP Filter Profile

The SIP Filter Profile is a collection of the configurable filter settings of individual SIP headers. Depending on the filter settings of each of the SIP headers in the SIP Filter Profile, the SBC either relays the SIP messages without parsing the header, or parses the headers of the messages.

For every SIP message associated with the ingress leg of the call, the SBC first checks the SIP Filter Profile for the filter setting of the SIP header. If the SIP Filter Profile indicates that a particular SIP header needs filtering, the SBC stores it without parsing.

The SIP header of the egress leg is populated on the basis of the configuration in the IP Signaling Profile and the Transparency Profile. In the egress leg of a call, the transparency bit mask is set to identify the headers that are transparently passed. If the transparency settings of all unknown headers in the IP Signaling Profile is enabled, all the stored headers (including the ones filtered in the ingress leg), is copied to the SIP header of the egress leg.

If the Transparency Profile attached with the Egress Trunk Group indicates that specific headers are allowed to pass transparently, and those headers are present as filtered headers, they are individually copied to the SIP header of the egress leg. In this case, the transparency bits are enabled, either by the IP Signaling Profile or through the flexible header transparency.

**Note**

A filtered header of a call in a Gateway-Gateway (GW-GW) scenario is not passed to the egress gateway if that header is known to the egress gateway, even if the "unknown" header transparency is enabled.

For example, the P-Asserted-Identity (PAI) header is not passed transparently if the following events occur simultaneously:

- the PAI is filtered on the ingress
Configuring a SIP Filter Profile

Perform the following:

- Creating a new SIP Filter Profile
- Changing the Transparency Setting of a Header Under a SIP Filter Profile
- Configuring the Zone ID for the Selected Zone
- Configuring the Media IP Interface Group Name for a SIP Trunk Group
- Attaching the SIP Filter Profile to the SIP Trunk Group

Creating a new SIP Filter Profile

```bash
set profiles signaling sipFilterProfile doc_FILTER2
```

Changing the Transparency Setting of a Header Under a SIP Filter Profile

```bash
set profiles signaling sipFilterProfile doc_FILTER2 header Also enabled
```

Note

The SIP Filter Profile is not used for the egress leg of a call.

Note

The mandatory headers which are not part of the `sipFilterProfile` are as follows:

- From
- To
- Via
- Call-id
- Cseq
- Max-Forwards

Note

Avoid filtering headers pertaining to SIP call routing/protocol processing, as it may cause unexpected results such as call failures. Some of the headers that should not be filtered are as follows:

- Contact
- RSeq
- R Ack
- Record-Route
- Route
- Content-Type
- Content-Length
- WWW-Authenticate

Note

- the "unknown" header transparency is enabled on the egress gateway
- the PAI header is known to the egress gateway.

To pass the PAI header transparently on the egress gateway, enable transparency for the PAI header through a SIP Filter Profile.
Configuring the Zone ID for the Selected Zone

```
set addressContext default zone doc_ZONE_IAD id 10
```

Configuring the Media IP Interface Group Name for the SIP Trunk Group

```
set addressContext default zone doc_ZONE_IAD sipTrunkGroup doc_SBX10_IAD media
mediaIpInterfaceGroupName LIF1
```

Attaching the SIP Filter Profile to the SIP Trunk Group

```
set addressContext default zone doc_ZONE_IAD sipTrunkGroup doc_SBX10_IAD signaling sipFilterProfile
doc_FILTER2
```

### Note
- In the SIP Filter Profile, the default filter setting of all the headers is disabled. To filter a particular SIP header, change the setting to enabled.
- The SBC supports attaching a SIP Filter Profile to a SIP Trunk Group. If no SIP Filter Profile is configured for a SIP Trunk Group at the time of its creation, the default SIP Filter Profile is attached with the SIP Trunk Group.
- Even if the filter setting for an unknown header is set to disabled, the SBC transparently relays the SIP messages containing that header in the following cases:
  - The flexible header transparency indicates that the particular header should be transparently relayed.
  - The IP Signaling Profile indicates that the particular header should be transparently relayed. To transparently pass all unknown headers, set the transparency of the unknown headers to enabled in the IP Signaling Profile.

#### 3.11 Privacy Transparency

A new flag `anonymizeHostIpAddress` is introduced for the privacy parameter of the IP Signaling Profile to enable or disable this feature. When this flag is activated, the SBC anonymizes the incoming host IP portion of the private headers (P-AID, P-PID, RPID) by replacing it with the IP address of the SBC, before sending it to the egress leg of a call.

This feature supports anonymizing the host IP address for the following message types:

- INVITE
- BYE
- OPTIONS
- SUBSCRIBE
- NOTIFY
- UPDATE
- PUBLISH
- MESSAGE

### Note
This feature currently supports only in-dialog messages, and not out-of-dialog messages.

Anonymizing the Host Ip Address
Perform the following:

- Enabling the transparency Flag
- Enabling the anonymizeHostIpAddress Flag

**Enabling the transparency Flag**

To enable the transparency flag for an ipSignalingProfile, enter the following command:

```
set profiles signaling ipSignalingProfile DEFAULT_SIP egressIpAttributes privacy transparency enable
```

**Enabling the anonymizeHostIpAddress Flag**

To enable the anonymizeHostIpAddress flag to activate this feature, enter the following command:

```
set profiles signaling ipSignalingProfile DEFAULT_SIP egressIpAttributes privacy anonymizeHostIpPortion enable
```