SIP Endpoint SDK Developer's Guide

SIP Monitoring

12/14/2025

Contents

SIP Monitoring

> ## Important
> This feature was introduced as part of the 9.0.008.04 release.

This feature allows custom application to get SDP content and negotiated codec name for the currently active session, and to get SIP messages for active or recently released session.

OS X

The following methods have been added to the GSSessionService:

```
/**
 Session service protocol
 */
@protocol GSSessionService <NSObject>
...
/**
 Gets codec name for active or held session.
 @param sessionId active or held session ID.
 @returns the code name.
 */
- (NSString*) getCodecNameForSessionId:(int) sessionId;
/**
 Gets last remote offer or answer SDP for active or held session.
 @param sessionId active or held session ID.
 @returns SDP content.
 */
- (NSString*) getRemoteSDPForSessionId:(int) sessionId;
/**
 Gets specific SIP message from history by its position index.
 @param sessionId active or released session ID.
 @param index position in SIP history.
 @returns SIP message content.
 */
- (NSString*) getSIPMessageForSessionId:(int) sessionId
                                  byIndex:(int) index;
/**
 Gets specific SIP message from history by its name.
 @param sessionId active or released session ID.
 @param name SIP message name.
 @returns SIP message content.
 */
- (NSString*) getSIPMessageForSessionId:(int) sessionId
                                   byName:(NSString*) name;
```

.NET

The following methods have been added to the ICallControl interface:

```
public interface class ICallControl {
...
  String^ GetCodecName (int sessionId);
  String^ GetRemoteSDP (int sessionId);
  String^ GetSIPMessage(int sessionId, int index);
  String^ GetSIPMessage(int sessionId, String^ name);
}
```

Detailed Description

All methods gets integer session ID (of the session queried) as the first parameter, which is the same ID as used in all session-related methods and which should be taken from session object by retrieving:

- property "callId" in SDK for OS X (note that, for historical reasons, property named "sessionId" holds different value - namely, the value of SIP header Call-ID for given session)

- property "SessionIntId" in SDK for .NET

Return value is a string representation of requested value, as described below, or empty string if the operation cannot be performed of requested data is not available.

Getting codec name

Corresponding method returns canonical name of currently negotiated codec for given session (e.g. "pcmu/8000" or "opus/48000/2")

Getting remote SDP content

Corresponding method returns last remote SDP received for given session as text

Getting SIP messages

Two methods are provided to obtain specific SIP message for given session, using either:

- message index (starting from 1) in the sequence of all SIP messages sent or received for this session,

- SIP message name, to get the last SIP message for this session with matching name

For the second approach, name parameter may be either SIP method (e.g. "INVITE" or "BYE"), SIP response code as string (e.g. "200") or special name "1xx" that matches any provisional response. In addition, the value may be prefixed with "<<" (to consider only incoming message) or "">>" (for outgoing messages only). For example:

- "INVITE" = last INVITE message (sent or received)

- "<<INVITE" = last incoming INVITE
- ">>ACK" = last ACK sent by endpoint

Return value includes full text of SIP message, including all headers and message body, or empty string if the requested message cannot be found.

Code Samples

OS X

The OSX code sample is written in the assumption that application code uses class with 'ep' property referring to GSEndpoint object. That is, it includes the following in the app header:

```
@property (nonatomic, retain) id<GSEndpoint> ep;
```

and the following initialization code:

```
self.ep = [GSEndpointFactory initSipEndpoint:configData];
```

Copying all SIP messages for given session into string array

This code fragment copies all SIP messages for session with given ID of callId into string array:

```
NSMutableArray *sipMessages = [[[NSMutableArray alloc] init] autorelease];
for (int i = 1;; i++) {
  NSString *msg = [[self.ep sessionControlService] getSIPMessageForSessionId:callId
byIndex:i];
  if (msg.length == 0) break;
  [sipMessages addObject:msg];
}
```

> **Important**
>
> Index starts at 1 (not 0) and the condition for end of list is returning empty value.

Printing session info to log

The following code prints some session information into log, including codec name, remote SDP, and "last incoming INVITE":

```
NSLog(@"Negotiated codec is %@", [[self.ep sessionControlService]
getCodecNameForSessionId:callId]);
NSLog(@"Remote SDP is %@",       [[self.ep sessionControlService]
getRemoteSDPForSessionId:callId]);
NSLog(@"Last incoming INVITE is %@", [[self.ep sessionControlService]
                                      getSIPMessageForSessionId:callId
byName:@"<<INVITE"]);
```

.NET

The .NET code sample is written in the assumption that the application code uses class with 'endpoint' property:

```
private SipEndpoint.IEndpoint endpoint;
```

and the following initialization code:

```
this.endpoint = SipEndpoint.EndpointFactory.CreateSipEndpoint();
this.callControl = this.endpoint.Resolve("ICallControl") as CallControl;
```

Copying all SIP messages for given session into string array

This code fragment copies all SIP messages for session with given ID of callId into string array:

```
string[] sipMessages = new string[100];
for (int i = 1; i<100; i++) {
  string msg = this.callControl.GetSIPMessage(callId, i);
  if (msg.Length == 0) break;
  sipMessage[i] = msg;
}
```

> ### Important
> Index starts at 1 (not 0) and the condition for end of list is returning empty value.

Printing session info to log

The following code prints some session information into log, including codec name, remote SDP, and "last incoming INVITE":

```
this.extendedService.LogPrint(4,"Negotiated codec is
"+this.callControl.GetCodecName(callId));
this.extendedService.LogPrint(4,"Remote SDP is "
+this.callControl.GetRemoteSDP(callId));
this.extendedService.LogPrint(4,"Last incoming INVITE is "
                                    +this.callControl.GetSIPMessage(callId,
"<<INVITE"));
```