SIP Endpoint SDK Developer's Guide

SIP Endpoint SDK 8.1.2

2/11/2022

Table of Contents

Welcome

Warning

This documentation is outdated and is included for historical purposes only.

This Developer's Guide contains information that will help you understand:

- The architecture of the SIP Endpoint SDK
- Its configuration options
- How to design applications that make use of the powerful features of this SDK

Before reading this material you may want to:

- Install the SIP Endpoint SDK on your .NET or OS X system.
- Ensure you have access to the latest version of the API Reference.
- Download the latest version of the Release Note (using links on the SIP Endpoint SDK Product Page) to see the most recent news and updates about this product.

Developing with SIP Endpoint SDK for .NET

Warning

This documentation is outdated and is included for historical purposes only.

This chapter shows how you can develop applications using the SIP Endpoint SDK for .NET.

Default SipEndpoint.config Settings

Warning

This documentation is outdated and is included for historical purposes only.

Using the Default Configuration File

You can find the default configuration file in the following location:

```
<installation folder>/Configuration/SipEndpoint.config
```

This file contains XML configuration details that affect how your SIP Endpoint SDK application behaves. The inital settings are the same as those specified for use with the QuickStart application that is included with your SIP Endpoint SDK release.

Configuration settings are separated into two containers: the Basic Container holds the connectivity details that are required to connect to your SIP Server, while the Genesys Container holds a variety of configuration settings.

Basic Container

The first Container ("Basic") holds the basic connectivity details that are required to connect to your SIP Server. This container has at least one connection (Connectivity) element with the following attributes:

```
<Connectivity user="DN" server="SERVER:PORT" protocol="TRANSPORT"/>
```

If you are using a configuration that supports Disaster Recovery and Geo-Redundancy, there may be multiple connection elements present with each specifying a separate possible connection. Refer to the configuration settings of that feature for details. You will have to make the following changes and save the updated configuration file before using the SIP Endpoint SDK:

- user="DN" — Supply a valid DN for the user attribute.
- server="SERVER:PORT" — Replace SERVER with the host name where your SIP Server is deployed, and PORT with the SIP port of the SIP Server host. (The default SIP port value is 5060.)
- protocol="TRANSPORT" — Set the protocol attribute to reflect the protocol being used to communicate with SIP Server. Possible values are UDP, TCP, or TLS.

Genesys Container

The second Container ("Genesys") holds a number of configurable settings that are organized into domains and sections. These settings do not have to be changed, but can be customized to take full control over your SIP Endpoint SDK applications.

An overview of the settings in this container and the valid values for these settings is provided here:

Section	Setting	Values	Description
codecs — See Working with Codec Priorities			
device			
	audio_in_device	String	Microphone device name
	audio_out_device	String	Speaker device name
	headset_name	String	The name of the headset model
	manual_audio_devices_configure	Number	Valid values: 0 or 1. The setting is active if `use_headset=0`. Enables configuration of the preferred input and output devices that are set in `audio_in_device` and `audio_out_device`.
	use_headset	Number	Valid values: 0 or 1. If set to 1, the SDK uses a headset as the preferred audio input and output device.
diagnostics			
	enable_logging	Number	Valid values: 0 or 1. Disable or enable logging.
	log_file	String	Log file name, for example, `SipEndpoint.log`
	log_level		Valid values: 0 – 4. Log levels: 0 = "Fatal"; 1 = "Error"; 2 = "Warning"; 3 = "Info"; 4 = "Debug".
	log_options_provider	String	Valid values for webrtc = (warning, state, api, debug, info, error, critical). For example: `gsip=2, webrtc=(error,critical)`
	logger_type	`file`	If set to `file`, the log data will be printed to the file specified by the `log_file` parameter.

Section	Setting	Values	Description
endpoint			
	audio_qos	Number	The integer value representing the DSCP bits to set for RTP audio packets. **Note:** QoS is not supported for Windows Vista, Windows 7, or higher.
	include_os_version_in_user_agent_header	Number	If set to 1, the user agent field includes the OS version the client is currently running on. Default: 0.
	ip_versions	IPv4 IPv6 IPv4,IPv6 IPv6,IPv4 empty	A value of `IPv4` means that the application selects an available local IPv4 address; IPv6 addresses are ignored. A value of `IPv6` means that the application selects an available local IPv6 address; IPv4 addresses are ignored. A value of `IPv4,IPv6` or an empty value means that the application selects an IPv4 address if one exists. If not, an available IPv6 address is selected. A value of `IPv6,IPv4` means that the application selects an IPv6 address if one exists. If not, an available IPv4 address is selected. Default: `IPv4,IPv6`. NOTE: This parameter has no effect if the `public_address` option specifies an explicit IP address.
	public_address	String	Local IP address or Fully Qualified Domain Name (FQDN) of the machine.
	rtp_inactivity_timeout	Number	Timeout interval for RTP inactivity. Valid values are integers from `0` to `150`. A value of `0` or values greater than `150` mean that this feature is not activated. A value in the range of `1` to `150` indicates the inactivity timeout interval in seconds. Default: `0`.
	rtp_port_min	Number	The integer value representing the minimum value for an

Section	Setting	Values	Description
			RTP port range. Must be within the valid port range of `9000` to `65535`. If the minimum and maximum values are not specified or are set to an invalid value, the default minimum (`9000`) and maximum (minimum value + 999) are used. Setting the minimum to a value that is larger than the maximum is considered an error and will result in a failure to initialize the endpoint.
	rtp_port_max	Number	The integer value representing the maximum value for an RTP port range. Must be within the valid port range of `9000` to `65535`. If the minimum and maximum values are not specified or are set to an invalid value, the default minimum (`9000`) and maximum (minimum value + 999) are used. Setting the maximum to a value that is less than the minimum is considered an error and will result in a failure to initialize the endpoint.
	signaling_qos	Number	The integer value representing the DSCP bits to set for SIP packets. **Note:** QoS is not supported for Windows Vista, Windows 7, or higher.
	sip_port_min	Number	The integer value representing the minimum value for a SIP port range. Must be within the valid port range of 1 to 65535. If the minimum and maximum values are not specified or are set to an invalid value, the

Section	Setting	Values	Description
			default minimum (`5060`) and maximum (minimum value + 6) are used. Setting the minimum to a value that is larger than the maximum is considered an error and will result in a failure to initialize the endpoint.
	sip_port_max	Number	The integer value representing the maximum value for a SIP port range. Must be within the valid port range of 1 to 65535. If the minimum and maximum values are not specified or are set to an invalid value, the default minimum (`5060`) and maximum (minimum value + 6) are used. Setting the maximum to a value that is less than the minimum is considered an error and will result in a failure to initialize the endpoint.
	video_qos	Number	The integer value representing the DSCP bits to set for RTP Video packets. **Note:** QoS is not supported for Windows Vista, Windows 7, or higher.
	vq_report_collector		See Producing RTCP Extended Reports
	vq_report_publish		See Producing RTCP Extended Reports
mailbox			
	password	String	Mailbox password
	server	String	Proxy server address and port for this mailbox
	timeout	Number	Registration timeout interval
	transport	udp tcp tls	Transport protocol to use when communicating with server

Section	Setting	Values	Description
	user	String	User ID for this mailbox
nat			
	ice_enabled	Boolean	Enable or disable ICE
	stun_server	String	STUN server address. An empty or null value indicates this feature is not being used.
	stun_server_port	String	STUN server port value
	turn_password	Number	Password for TURN authentication
	turn_relay_type	Number	Type of TURN relay
	turn_server	String	TURN server address. An empty or null value indicates this feature is not being used.
	turn_server_port	String	TURN server port value
	turn_user_name	String	User ID for TURN authorization
proxy<*n*>			
	display_name	String	Proxy display name
	password	String	Proxy password
	reg_interval	Number	The period, in seconds, after which the endpoint starts a new registration cycle when a SIP proxy is down. Valid values are integers greater than or equal to `0`. If the setting is empty or negative, the default value is `0`, which means no new registration cycle is allowed. If the setting is greater than `0`, a new registration cycle is allowed and will start after the period specified by `regInterval`.
	reg_timeout	Number	The period, in seconds, after which registration should expire. A new REGISTER request will be sent before expiration. Valid values are integers greater than or equal to `0`. If the setting is `0` or `empty/null`, then registration

Section	Setting	Values	Description
			is disabled, putting the endpoint in standalone mode.
security			
	cert_file	String	Thumbprint value of the Public endpoint certificate file, which is used as a client-side certificate for outgoing TLS connection and server-side certificate for incoming TLS connections. For example: `78 44 34 36 7a c2 22 48 bd 5c 76 6b 00 84 5d 66 83 f5 85 d5`
	tls_enabled	Number	If set to 1, connection with TLS transport will be registered. Default: `0`.
	use_srtp	String disabled optional mandatory	Indicates whether to use SRTP
session			
	agc_mode	0 1	If set to 0, AGC (Automatic Gain Control) is disabled; if set to 1, it is enabled. Default: 1. Other values are reserved for future extensions. This configuration is applied at startup, after which time the `agc_mode` setting can be changed to 1 or `0` from the main sample application. NOTE: It is not possible to apply different AGC settings for different channels in multi-channel scenarios.
	auto_accept_video	Number	If set to 1, video calling is enabled and if set to 0, video calling is disabled
	auto_answer	Number	If set to 1, all incoming calls should be answered automatically.

Section	Setting	Values	Description
	dtmf_method	Rfc2833 Info InbandRtp	Method to send DTMF
	dtx_mode	Number	Valid values: 0 or 1. If set to 1, DTX is activated.
	reject_session_when_headset_na	Number	Valid values 0 or 1. If set to 1, the SDK should reject the incoming session if a USB headset is not available.
	sip_code_when_headset_na	Number	If a valid SIP error code is supplied, the SDK rejects the incoming session with the specified SIP error code if a USB headset is not available.
	vad_level	Number	Sets the degree of bandwidth reduction. Valid values: 0 – 3 — from 0 (conventional VAD) to 3 (aggressive high).

Additional Configuration Options

The default configuration file may not contain all settings that may be used with the SIP Endpoint SDK; additional settings can be added to change certain behaviors. Check Configuring SIP Endpoint SDK for .NET for a discussion of these additional settings.

Configuring SIP Endpoint SDK for .NET

Warning

This documentation is outdated and is included for historical purposes only.

Basic Configuration Settings

You may want to set certain basic SIP Endpoint configuration options in your application. The following sections contain information that may be helpful in doing that.

Most configuration settings described on this page can be found and changed in the SipEndpoint.config file that is included with your SIP Endpoint SDK installation. For more information about this file, see Default Config Settings.

Timeout Issue

Note that the `reg_timeout` setting is valid only if it contains a numeric value. If the supplied value is a non-numeric value or is empty or null, it will be interpreted as `0` and the endpoint will operate in standalone mode.

IPv6

As of release 8.1.2, SIP Endpoint SDK for .NET supports TCP/IP version 6 (IPv6) on Windows Vista and higher.

The local address must be set to IPv6:

- By means of an explicit address specification (as a hostname resolving to IPv6 only, or by the use of a literal IP address)
- By setting the `ip-versions` option to favor IPv6
- Or by only having an IPv6 interface available on the host

The SIP proxy address must be either a hostname that primarily resolves to IPv6 or a literal IPv6 address. Note that the preferences in the `ip-versions` option do not currently affect external address resolution.

Note: IPv6 must be configured on both ends of a session — mixed IPv4 and IPv6 environments are not supported.

This feature is resolved at Endpoint creation time by applying the following configuration to the SIP

Provider.

Settings:

```
<setting name="public_address" value="Address"/>
<setting name="ip_versions" value="ipv4,ipv6"/>
```

QoS Bits

You can mark RTP packets by setting the QoS (TOS) bits so that network routers can prioritize them. This can make a big difference in the quality of your voice transmissions. When the SIP Endpoint is initializing, it gets information on QoS (TOS) bit settings from the SipEndpoint.config file. This information cannot be changed while the endpoint is running. The `system:qos:audio` setting allows you to specify a string value that properly marks your RTP packets, and the type of QoS supported for audio. For example, you could specify the type of QOS with a value such as "tos 22". If QOS is not supported for audio, leave this setting empty.

Note: QoS is not supported for Windows Vista, Windows 7, or higher.

Diagnostics

In order to diagnose production issues, it is important to have access to diagnostic information that relates to your SIP endpoint and its environment. Because of this, the SIP Endpoint SDK provides logging with 5 possible logging levels.

Valid values for the SIP Endpoint SDK logging levels are:

```
Fatal = 0
Error = 1
Warning = 2
Info = 3
Debug = 4
```

```
<domain name="system">

</domain>
```

Headset Connectivity Notification

The SIP Endpoint SDK only supports headset monitoring when the device is explicitly defined in the `SipEndpoint.config` file, as shown below:

```
<domain name="policy">
...

</domain>
```

In this case, headset connectivity is fully supported and the application can receive events indicating that the device has been plugged or unplugged.

It is also possible to use your application to select audio devices automatically, by using the SIP Endpoint SDK API to look for IN and OUT devices. Using the `GetAllSystemAudioInDevices()` and `GetAllSystemAudioutDevices()` methods allows you to check for attached devices, which can then be used to update the following settings before initializing the SIP Endpoint SDK.

```
<domain name="policy">
...

</domain>
```

However, the plugged or unplugged states are not monitored for the selected device in this case. If no devices are defined in the `SipEndpoint.config` file, then the SIP Endpoint SDK uses a default procedure to determine what device is attached, and will not monitor for plugged or unplugged states.

SIP Messaging and Media Encryption

SIP Endpoint SDK has settings that allow you to encrypt SIP messaging and the media channel. The SIP messaging can be encrypted using TLS, while the media channel can be encrypted using SRTP. The following diagram shows the SIP Endpoint SDK architecture with these features enabled.

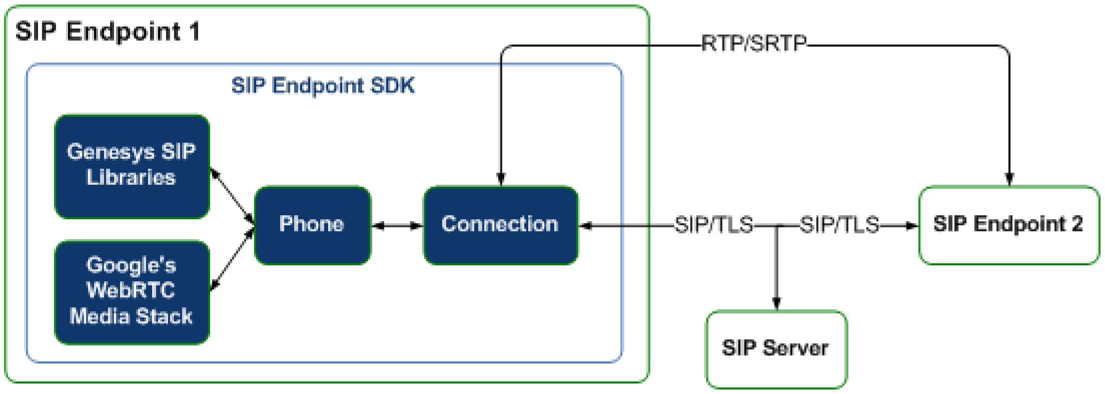


Figure 1: SIP Endpoint SDK Architecture with TLS and SRTP Enabled

To enable TLS support, set the protocol to `tls`, as shown here. The outgoing SIP messaging will be encrypted:

```
<Container name ="Basic">
  <Connectivity user ="DN" server="SipServerHost:Port" protocol="tls"/>
</Container>

<domain name="system">
. . .

</domain>
```

Use the `use_srtp` setting to control media channel encoding. Valid values are shown in the above code sample.

Digest Authentication Support

SIP Endpoint SDK supports the RFC2617-style digest authentication that is currently used by SIP Server. This authentication is triggered by using the following DN configuration options: `password` and `authenticate-requests`. If these options are configured for a specific DN, then SIP Server enforces the authentication mechanism for that DN. During registration, the SIP Endpoint receives a `401 Unauthorized` SIP challenge message request from SIP Server. The SIP Endpoint should then provide the encrypted password, along with the `Digest username`, in the next `REGISTER` message.

At this point, SIP Server compares the received password to the password associated with the DN object defined in Config Server. If the passwords match, then the SIP Endpoint can be registered and may proceed to access the call functionality provided by SIP Server. SIP Endpoint SDK provides the following method to set the password for a particular connection ID. This method has been added to the `IExtendedService` interface in the `Genesyslab.Sip.Endpoint.Provider.Genesys` namespace:

```
void SetPassword(int connectionId, String^ password);
```

Producing RTCP Extended Reports

You can use SIP Endpoint SDK to produce RTCP Extended Reports (RFC 3611) and publish them according to RFC 6035 at the end of each call, using a collector address of your choice.

Note: The publish message is sent to the specified collector address only if the `vq_report_collector` parameter is configured with the `user@server:port` format. For example, `collector@172.24.133.136:5160`.

Settings:

```
<domain name="policy">
<section name="endpoint">
...
<!-- (int) publish VQ report: 0=never, 1=end-of-call -->
<setting name="vq_report_publish" value="0"/>
<!-- (str) collector address (if NULL/empty => publish to proxy) -->
<setting name="vq_report_collector" value="vq_report_collector_URI"/>
</section>
```

Endpoint:

The `vq_report_publish` and `vq_report_collector` settings can be read from the Endpoint Policy by using the following methods:

```
GetEndpointPolicy(EndpointPolicyQuery.VqReportCollector);
GetEndpointPolicy(EndpointPolicyQuery.VqReportPublish);
```

Audio Layer Selection

SIP Endpoint SDK 8.1.2 now allows you to select an audio layer for WebRTC using the Windows environment variable `GCTI_AUDIO_LAYER`. If this variable is set to 1, SIP Endpoint SDK will use the Windows Wave audio layer. Otherwise, it will use the Windows Core audio layer. You can set this variable at Computer -> Properties -> Advanced System Settings -> Environment Variables.

Other Items for Review

The following items have been placed on this page in order to expedite their review. They will be moved to an appropriate page prior to publication.

SIP::INFO Message Exchange

SIP Endpoint SDK can create and transmit In-Dialog `SIP:INFO` messages using updated `Content-`

Type and `Content` headers. It can also receive messages with customer-provided information in the `Content-Type` and `Content` headers.

The following method has been added to the `ICallControl` interface in order to create and transmit an In-Dialog `SIP:INFO` message::

```
void SendSipInfo(int sessionId, String^ contentType, String^ content);
```

The following event has been added to Session Manager:

```
public event EventHandler<SessionEventArgs> SipInfoReceived;
```

The `SessionEvent` properties now also have these key value pairs:

```
Properties["ContentType"].ToString();
Properties["Content"].ToString();
```

RTP Statistics for .NET

SIP Endpoint SDK for .NET now allows you to monitor RTP video and audio statistics.

Methods have been added to the `ICallControl` interface so you can get audio and video statistics, both during a session and when the session has been disconnected:

```
Dictionary<String^,Object^>^ GetAudioStatistics(int sessionId);
Dictionary<String^,Object^>^ GetVideoStatistics(int sessionId);
```

The properties structure of these statistics is the same for both audio and video, as shown here:

```
Dictionary<String^,Object^>^ properties;

// positive if next 5 fields are valid (local stat available)
properties->Add("Got Local Stat ". . .);
// fraction of lost packets (i.e. not locally received)
properties->Add("Local Frac Lost". . .);
// interarrival jitter
properties->Add("Local Jitter   ". . .);
// local octet count (number of bytes sent)
properties->Add("Local Oct Count". . .);
// local packet count (number of RTP packets sent from our side)
properties->Add("Local Pkt Count". . .);
// total lost packaged (calculated per RFC 3550)
properties->Add("Local TotalLost". . .);

// true if next 3 fields are valid (got Receiver Report from remote side)
properties->Add("Got Remote RR  ". . .);
// fraction of lost packets (i.e. not received by remote end)
properties->Add("Remote FracLost". . .);
// interarrival jitter
properties->Add("Remote Jitter  ". . .);
// total lost packaged (calculated per RFC 3550)
properties->Add("Remote TotaLost". . .);

// true if next 2 fields are valid (got Sender Report from remote side)
properties->Add("Got Remote SR  ". . .);
// remote octet count
properties->Add("Remote OctCount". . .);
// remote packet count (number of packets sent by remote side)
properties->Add("Remote PktCount". . .);
```

```
// Round-trip time in milliseconds.
properties->Add("Round Trip Time". . .);
```

SIP Endpoint SDK Disaster Recovery and Geo-Redundancy

Warning
This documentation is outdated and is included for historical purposes only.

This article describes the ways in which SIP Server SDK supports high availability and resilience.

Introduction

The overall architecture for the disaster recovery/geo-redundancy solution is depicted below. Using this architecture, the SIP Endpoint SDK can connect to multiple sites in different geographical locations, providing redundancy and the potential for quick and efficient disaster recovery.

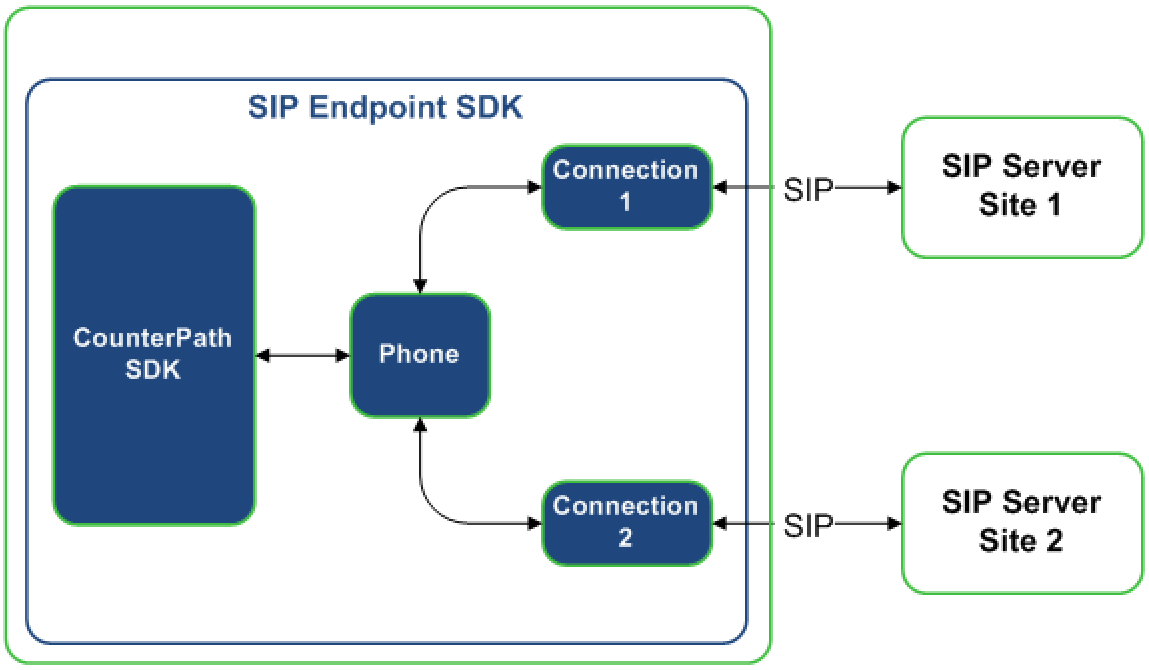


Figure 1: Geographical Redundancy and Disaster Recovery Architecture.

The SIP Endpoint SDK includes a QuickStart application that can be used to run functional testing. The QuickStart application starts SIP Endpoint SDK by creating a Phone instance and then using configuration settings to create two SIP connections and register them on two separate SIP Server sites.

To implement this type of architecture, you must use SIP Server release 8.1.0 or later.

How It Works

To enable Geo-Redundant architecture, the SIP Server SDK has been changed to allow multiple SIP Servers to be registered. Each SIP Server site requires a distinct connection to be made.

Connection Behaviors

Each connection created in the SIP Endpoint SDK has the following behavior:

SIP Server State	Connection Behavior
Available	Registers and works with the SIP Server normally.
Down or Unavailable	Continually attempts to register until the connection is either successful or explicitly removed. Once a specified SIP Server comes up after being down, then the SIP Endpoint SDK registers and works with the SIP Server normally as soon as it becomes available.

SIP Endpoint SDK will register separate connections on both SIP Server sites - even if they are associated with DNs that have the same name. On every registration renewal attempt, the SIP Endpoint SDK checks for successful registration to determine if that SIP Server is down or unavailable. Whenever there is a change in status for a connection, the SIP Server SDK generates a notification event that is sent to the application with information about the connection status. When the SIP Endpoint SDK is using multiple connections, equal priority is given to each connection. Registration on primary and secondary sites occurs simultaneously.

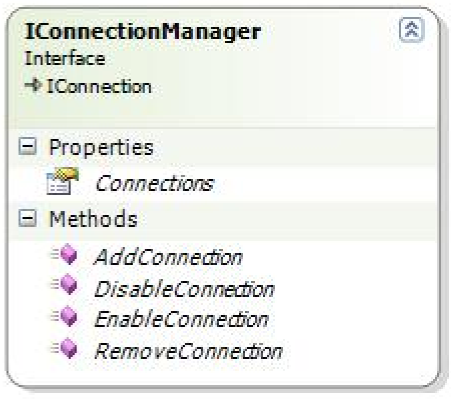
Working with Multiple Connections

When working with multiple connections, there are a few key items you should keep in mind:

- 1pcc outbound calls use a particular SIP Server site by dialing from an explicitly assigned connection. All other calls are put on hold. If the intended SIP Server site is down then the 1pcc outbound call is made from the second site.
- 3pcc inbound calls are received and answered using the corresponding connection, while all other calls are put on hold. (Note: The 2nd line is ringing while 1st line is on call.) If the first site is down or unavailable then the SIP Endpoint SDK receives 3pcc inbound calls by using second connection.
- If a call disconnection notification is received, check the reason why call was disconnected along with the connection ID. A call disconnection reason of "SipError" along with the message "408" or "Request Timeout" can be interpreted as a sign that the corresponding SIP Server is down or unavailable.

SIP Endpoint SDK API changes

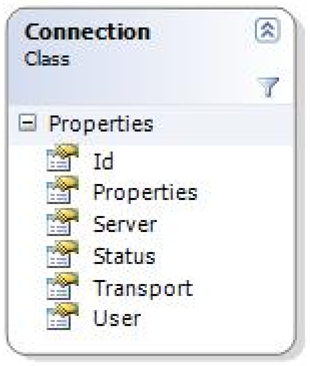
Handling Multiple Connections



To support handling more than one connection, the following methods have been added to IConnectionManager interface class that is implemented by CpProvider and ConnectionManager:

Description	Method Signature
Add a connection	void AddConnection(int connectionId);
Enable a connection	void EnableConnection (int connectionId);
Disable a connection	void DesableConnection (int connectionId);
Remove a connection	void RemoveConnection (int connectionId);
Get connections collection	ConnectionCollection Connections { get; }

Properties that can be accessed from the collection of connections are shown in the following class diagram:

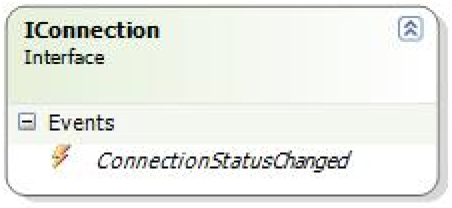


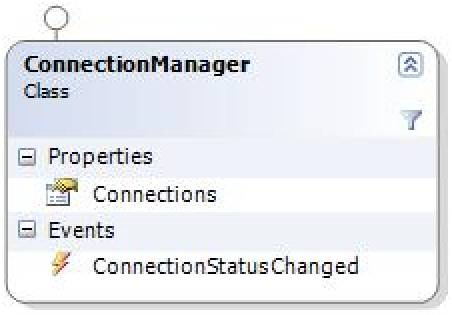
Managing Connection Status

Notification of changes to the connection status is given by the ConnectionStatusChanged event.

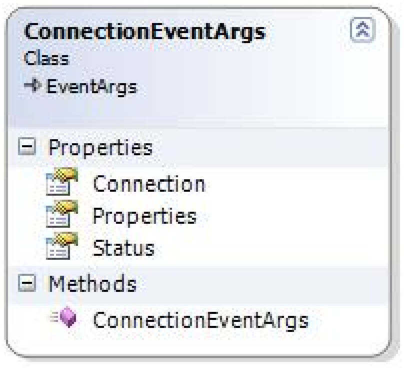
```
event EventHandler<ConnectionEventArgs> ConnectionStatusChanged;
```

This event has been added to IConnection interface as shown below:





The event includes ConnectionEventArgs, which provides your application access to the following properties:



Finding the Connection ID in Session Manager

The SessionStatusChanged event in session manager now includes a *connectionID* property in the *SessionEventArgs.Session.Properties* collection, indicating which connection is available.

Making Calls Using First-Party Call Control

The *Dial* method of the *ICallControl* interface now includes a *connectionID* parameter that determines which connection will be used for outgoing 1pcc calls.

```
void Dial(int connectionId, String^ destination);
```

Configuration Settings

In addition to the basic configuration details you should be aware of, the following settings must be added to your SipEndpoint.config file to support Disaster Recovery and Geo-Redundancy.

Adding Multiple Connections

To support configuring more than one connection, the section: <Container name ="Basic"> is extended to have more than one Connectivity tag. See the following example for details:

```
<Container name ="Basic">
    <Connectivity user ="DN1" server="SipServer1:port1" protocol="Protocol1"/>
    <Connectivity user ="DN2" server=" SipServer2:port2" protocol=" Protocol2"/>
</Container>
```

Where:

- DN1 and DN2 are extension objects in CME
- SipServer1:port1 and SipServer2:port2 are SIP Server host names and ports
- Protocol1 and Protocol2 are supported transport protocols by SIP Endpoint SDK

Changing the Re-Registration Timeout Interval in SIP Endpoint SDK

The default SIP Endpoint SDK re-registration interval is 3600 seconds. However, you can redefine that interval by updating the SipEndpoint.config file to include the reregister_in_seconds setting. The example shown below demonstrates how different re-registration intervals can be specified for two connections, and the new configuration applied to your SIP Endpoint SDK applications. In this example, proxy0 and proxy1 are equivalent to the SIP Endpoint SDK connections with connection ID 0 and 1.

```
<domain name="proxies">
    <section name="proxy0">
        ...
        <setting name="reregister_in_seconds" value="10"/>
    </section>
    <section name="proxy1">
        ...
        <setting name="reregister_in_seconds" value="20"/>
    </section>
</domain>
```

SIP Endpoint SDK .NET QuickStart Application

Warning

This documentation is outdated and is included for historical purposes only.

The easiest way to start using the SIP Endpoint SDK is with the bundled QuickStart application. This application ships in the SDK folder and is supplied as both an executable and as source code in the form of a Visual Studio project. You will need to enter environmental information in an XML configuration file before you can execute the application. After doing that, you can build and run the QuickStart application right away.

Configuring the QuickStart Application

Find either the `<SIP Endpoint SDK Folder>\QuickStartExe` folder or the `<SIP Endpoint SDK Folder>\QuickStart\\QuickStartWinFormVS9` folder and open the `SipEndpoint.config` file. The first few lines of the file look like this:

```
<?xml version="1.0" encoding="utf-8" ?>
<SipEndpoint xmlns="http://schemas.genesyslab.com/2009/sipendpoint">
  <Container name ="Basic">
    <Connectivity user ="8042" server="SEPSDK01.us.int.genesyslab.com:5060" protocol="udp"/>
  </Container>
  <Container name ="Genesys">
...
```

The first Container element is called "Basic," as you can see, and contains elements that you must supply values for. Otherwise, the QuickStart application will not work. Here are the things you need to do:

- Supply a valid DN for the user attribute.
- Replace SERVER in the server attribute with the name of the host where SIP Server is deployed.
- Replace PORT in the server attribute with the SIP port of the SIP Server host. (The default value is 5060.)

The Configuration page includes information on how to configure various settings, including RTP port ranges, QoS bits, diagnostics and endpoint DNs. These additional configuration options are not required to run the SIP Endpoint QuickStart application, but allow you to customize useful features for the created endpoints.

Running the QuickStart Application

To run the QuickStart application, you can either run the executable version by carrying out these steps:

1. Open the <SIP Endpoint SDK Folder>\QuickStartExe folder.
2. Double-click QuickStartWinFormVS9.exe.

Or you can build and run the Visual Studio project:

1. Open the <SIP Endpoint SDK Folder>\QuickStart\QuickStartWinFormVS9 folder.
2. Double-click QuickStartWinFormVS9.csproj.
3. Build the project.
4. Run the project.

The SIP Endpoint Interface

The QuickStart application demonstrates how to use the `IEndpoint` interface, which you should use in your own applications. As shown in the following code snippet from the QuickStart application, you use the `CreateSipEndpoint` method of the `EndpointFactory` class to create the endpoint:

```
[C#]
IEndpoint endpoint;
endpoint = EndpointFactory.CreateSipEndpoint(logger);
```

Note: Because the `CreateSipEndpoint` method creates a singleton, creation of more than one endpoint instance is not supported by Genesys. Other than that, most of the QuickStart application involves reading configuration values and updating the status of the endpoint. The real work is done by the T-Lib code that will reside elsewhere in your application.

SIP Endpoint SDK Video Support

> **Warning**
> **This documentation is outdated and is included for historical purposes only.**

This article discusses the video support that has been added to SIP Endpoint SDK. This support consists of:

- A new video control interface
- New video call control features that have been added to the existing call control interface

Video Control

The `IVideoControl` interface is in the `Genesyslab.Sip.Endpoint.Provider.CP` namespace. It allows you to work with events and information related to:

- Video source
- Frame events
- Window handles
- Connectivity mode
- Connectivity mode settings

These features are discussed in the following sections.

Video Source

This SDK provides methods to get or set video source–related information:

Method	Description
bool HasCaptureDevice();	Returns true if the SDK has been configured to work with a camera.
String^ GetCurrentCaptureDeviceName();	Returns the names of the devices currently being used for video.
array<VideoDevice^>^ GetAllSystemCaptureDevices();	Returns the names of all available cameras known to the operating system.
GsStatus SetCaptureDevice(int deviceId);	Picks up a capture device by its device ID and gets its status if the device has been successfully

Method	Description
	reserved.
VideoCapability^ GetVideoCapability(int deviceId, int N);	Returns capture device video capability by device ID and capability order number.
int GetNumberVideoCapabilities(int deviceId);	Returns a number of supported capabilities by device ID.

Video Frame Events

The SDK also provides video frame–related events:

Method	Description
event EventHandler<EndpointEventArgs^>^ VideoFrameSizeReceived;	Provides notification about video frame size.
event EventHandler<EndpointEventArgs^>^ VideoFrameDelivered;	Provides notification that a video frame has been received with size shown by `VideoFrameSizeReceived`.

Video Call Control == SIP Endpoint SDK's `ICallControl` interface, which is in the `Genesyslab.Sip.Endpoint.Provider.Genesys` namespace, has methods to dial and answer a session with or without video:

Method	Description
void Dial(int connectionId, String^ destination, bool video, String^ data);	Starts dialing to place an outgoing call from the connection with `ID = connectionId` to `destination`. If `video=true` the video is offered.
void Answer(int sessionId, bool video);	Answers an incoming call with `sessionId` and if `video=true` the offered video is accepted.

Access to Raw Video Frames

SIP Endpoint SDK 8.1.2 for .NET provides access to raw video frames in BGR32 format.

The following API has been added:

```
public enum class VideoRenderFormat {
        i420     = 0,
        YV12     = 1,
        YUY2     = 2,
        UYVY     = 3,
        IYUV     = 4,
        ARGB     = 5,
        RGB24    = 6,
        RGB565   = 7,
        ARGB4444 = 8,
        ARGB1555 = 9,
        MJPEG    = 10,
        NV12     = 11,
        NV21     = 12,
        BGRA     = 13,
        Unknown  = 99
};
```

You can use these methods to add and remove local and remote videos:

```
// Start/Stop Local Video to be shown in the Windows Form with handle
GsStatus StartLocalVideo(VideoCapability^ inOut, IntPtr handle,
         unsigned zOrder, float left, float top, float right, float bottom);
GsStatus StopLocalVideo();

// Start/Stop Remote Video for sessionId to be shown in the Windows Form with handle
GsStatus StartRemoteVideo(int sessionId, IntPtr handle, unsigned zOrder,
         float left, float top, float right, float bottom);
GsStatus StopRemoteVideo(int sessionId);
```

The following methods may be used to add or remove the local or remote video renderer:

```
// Add/Remove External Local Video Renderer
void AddLocalVideoRenderer(VideoRenderFormat format);
void RemoveLocalVideoRenderer();

// Add/Remove External Remote Video Renderer
void AddRemoteVideoRenderer(VideoRenderFormat videoFormat, int sessionId);
void RemoveRemoteVideoRenderer(int sessionId);
```

SIP Endpoint SDK Call Statistics

SIP Endpoint SDK Call Statistics

As of release 8.1.100.04 of SIP Endpoint SDK for .NET, two methods have been added to the `ICallControl` interface, so that you can have real-time access to RTP audio and video statistics during a call. These statistics are based on the RTCP packets that have been sent during a call session. The two methods are:

```
Dictionary<String^,Object^>^ GetAudioStatistics(String^ sessionId);
Dictionary<String^,Object^>^ GetVideoStatistics(String^ sessionId);
```

The NET QuickStart Application shows how to monitor statistics while a call is in progress and how to gather the most recent statistics before a call is disconnected.

RTCP is enabled by default.

Available Statistics

SIP Endpoint SDK provides video and audio statistics at the end of each call when the session achieves a status of `SessionStatus.Disconnected`. These statistics have the same names, meanings, and structure for both video and audio.

Local Statistics		
	Got Local Stat	This value is positive if the next 5 fields are valid (which means that local statistics are available)
	Local Frac Lost	The fraction of packets that were lost during this call (that is, packets that were not received by the local endpoint)
	Local Jitter	Interarrival jitter
	Local Oct Count	Local octet count (number of bytes sent)
	Local Pkt Count	Local packet count (number of RTP packets sent from the local endpoint)
	Local Total Lost	Total lost packets (calculated per RFC 3550)
Remote Receiver Report		
	Got Remote RR	This value is true if the next 3 fields are valid (which means that the local endpoint received a Receiver Report from the remote endpoint)
	Remote Frac Lost	The fraction of packets that were lost during this call (that is,

		packets that were not received by the remote endpoint)
	Remote Jitter	Interarrival jitter
	Remote Total Lost	Total lost packets (calculated per RFC 3550)
Remote Sender Report		
	Got Remote SR	This value is true if the next 2 fields are valid (which means that the local endpoint received a Sender Report from the remote endpoint)
	Remote Oct Count	Remote octet count
	Remote Pkt Count	Remote packet count (number of packets sent by the remote endpoint)
Round Trip		
	Round Trip Time	Round-trip time in milliseconds

Developing with SIP Endpoint SDK for Apple OS

Warning

This documentation is outdated and is included for historical purposes only.

This chapter shows how you can develop applications using the SIP Endpoint SDK for Apple OS.

Configuring SIP Endpoint SDK for Apple OS

Warning

This documentation is outdated and is included for historical purposes only.

The sample application that comes with the SIP Endpoint SDK for Apple OS distribution includes a property list (plist) that is used for configuring the application. This file is located at `<SIP Endpoint SDK Installation Folder>/Sample/Src/Sip EP Sample.plist`.

Note: Genesys recommends that you use the sample application as the starting point for your development efforts.

If you are developing applications from scratch, you should:

1. Create a copy of the plist from the sample application
2. Name it appropriately
3. Place it in your app's `Src` folder.

SIP Endpoint Configuration Settings

You can customize the following settings in your SIP Endpoint SDK applications.

Section	Setting	Values	Description
GSDefaultConnectionPolicy			
	networkInterface	String	Name of the network interface
GSDefaultDevicePolicy			
	audio_in_device	String	Microphone device
	audio_out_device	String	Speaker device
	use_headset	Boolean	If set to YES, the SDK uses a headset as the preferred audio input and output device.
GSDefaultEndpointPolicy			
	audioQos	Number	The integer value representing the DSCP bits to set for RTP audio packets.

Section	Setting	Values	Description
	includeOSVersionInUserAgentHeader	Boolean	If set to YES, the user agent field includes the OS version the client is currently running on. Default: `NO`.
	ip_version	IPv4 IPv6 IPv4,IPv6 IPv6,IPv4 empty	A value of `IPv4` means that the application selects an available local IPv4 address; IPv6 addresses are ignored. A value of `IPv6` means that the application selects an available local IPv6 address; IPv4 addresses are ignored. A value of `IPv4,IPv6` or an empty value means that the application selects an IPv4 address if one exists. If not, an available IPv6 address are selected. A value of `IPv6,IPv4` means that the application selects an IPv6 address if one exists. If not, an available IPv4 address are selected. Default: `IPv4,IPv6`. NOTE: This parameter has no effect if the `public_address` option specifies an explicit IP address. NOTE: Although the IPv6 configuration options are available in the plist file for this product, they are not supported in the 8.1.2 Release of SIP Endpoint SDK for Apple OS.
	public_address	String	Local IP address or Fully Qualified Domain Name (FQDN) of the machine. NOTE: Although the IPv6 configuration options are available in the plist file for this product, they are not supported in the 8.1.2 Release of SIP Endpoint SDK for Apple OS.
	rtpInactivityTimeout	Number	Timeout interval for RTP inactivity. Valid values are integers from `0` to `150`. A value of `0` or values greater than `150` mean that this feature is not activated. A value in the range of `1` to `150` indicates the inactivity timeout interval in seconds. Default: `0`.

Section	Setting	Values	Description
	rtpPortMin	Number	The integer value representing the minimum value for an RTP port range. Must be within the valid port range of `9000` to `65535`. If the minimum and maximum values are not specified or are set to an invalid value, the default minimum (`9000`) and maximum (minimum value + `999`) are used. Setting the minimum to a value that is larger than the maximum is considered an error and will result in a failure to initialize the endpoint.
	rtpPortMax	Number	The integer value representing the maximum value for an RTP port range. Must be within the valid port range of `9000` to `65535`. If the minimum and maximum values are not specified or are set to an invalid value, the default minimum (`9000`) and maximum (minimum value + `999`) are used. Setting the maximum to a value that is less than the minimum is considered an error and will result in a failure to initialize the endpoint.
	secureSignalingQos	Number	The integer value representing the DSCP bits to set for TCP packets.
	signalingQos	Number	The integer value representing the DSCP bits to set for SIP packets.
	sipPortMin	Number	The integer value representing the minimum value for a SIP port range. Must be within the valid port range of `1` to `65535`. If

Section	Setting	Values	Description
			the minimum and maximum values are not specified or are set to an invalid value, the default minimum (5060) and maximum (minimum value + 6) are used. Setting the minimum to a value that is larger than the maximum is considered an error and will result in a failure to initialize the endpoint.
	sipPortMax	Number	The integer value representing the maximum value for a SIP port range. Must be within the valid port range of 1 to 65535. If the minimum and maximum values are not specified or are set to an invalid value, the default minimum (5060) and maximum (minimum value + 6) are used. Setting the maximum to a value that is less than the minimum is considered an error and will result in a failure to initialize the endpoint.
	videoQos	Number	The integer value representing the DSCP bits to set for RTP Video packets.
GSDefaultSessionPolicy			
	AGC_mode	0 1	If set to 0, AGC (Automatic Gain Control) is disabled; if set to 1, it is enabled. Default: 1. Other values are reserved for future extensions. This configuration is applied at startup, after which time the `agc_mode` setting can be changed to 1 or `0` from the main sample application.

Section	Setting	Values	Description
			NOTE: It is not possible to apply different AGC settings for different channels in multi-channel scenarios.
	auto_accept_video	Boolean	If set to YES, all incoming video should be accepted automatically. NOTE: The video mode window will not be opened if `auto_accept_video` and `auto_answer` are both set to `0`.
	auto_answer	Boolean	If set to YES, all incoming calls should be answered automatically. NOTE: The video mode window will not be opened if `auto_accept_video` and `auto_answer` are both set to `0`.
	dtmf_method	Rfc2833 Info InbandRtp	Method to send DTMF
	reject_session_when_headset_na	Boolean	If set to YES, the SDK should reject the incoming session if a USB headset is not available.
	sip_code_when_headset_na	Number	If a valid SIP error code is supplied, the SDK rejects the incoming session with the specified SIP error code if a USB headset is not available.
basic: account connection details			
	regInterval	Number	The period, in seconds, after which the endpoint starts a new registration cycle when a SIP proxy is down. Valid values are integers greater than or equal to `0`. If the setting is empty or negative, the default value is `0`, which means no new registration cycle is allowed. If the setting is

Section	Setting	Values	Description
			greater than `0`, a new registration cycle is allowed and will start after the period specified by `regInterval`.
	registrationTimeout	Number	The period, in seconds, after which registration should expire. A new `REGISTER` request will be sent before expiration. Valid values are integers greater than or equal to `0`. If the setting is empty or negative, the default value is 1800 seconds. If the setting is `0`, registration is disabled, putting the endpoint in standalone mode.
	transport	udp tcp tls	The transport protocol to use when communicating with server
	server	String	The server address and port
	stun_server	String	STUN server address (with optional port). An empty or null value indicates this feature is not being used.
	turn_password	Number	Password for TURN authentication
	turn_server	String	TURN server address (with optional port). An empty or null value indicates this feature is not being used.
	turn_userName	String	User ID for TURN authorization
	user	String	User ID for this connection
basic: account mailbox details			
	server	String	Proxy server address and port for this mailbox
	timeout	Number	Registration timeout interval
	transport	udp	Transport protocol to use when

Section	Setting	Values	Description
		tcp tls	communicating with server
	user	String	User ID for this mailbox
codecs: priority			
	<Codec Name>	Number	Codec priority in SDP. A higher number means the codec has preference in codec negotiation.
diagnostics			
	enable_logging	Boolean	Enable or disable logging
	log_file	String	Log file name, for example, `SipEndpoint.log`
	log_level	debug info warn error fatal	Log levels
	Log_option_provider	#/Empty gsip=3,webrtc=(state,warning)	If set to `#/Empty`, log messages will not include webrtc messages. If set to `gsip=3,webrtc=(state,warning)` and the level is set to `info`, log messages will include webrtc messages.
	logger_type	file default	If set to `file`, the log data will be printed to the file specified by the `log_file` parameter. If set to `default`, the log data will be printed to the console.
security			
	ca_list_file	String	Certificate of Authority (CA) list file
	cert_file	String	Public endpoint certificate file, which is used as client-side certificate for outgoing TLS connection and server-side certificate for incoming TLS connection

Section	Setting	Values	Description
	method	unspecified tlsv1 sslv2 sslv3 sslv23	Security method
	password	String	Password to open private key
	privkey_file	String	Path to the optional private key file of the endpoint certificate to be used. Example: `/usr/local/ssl/certs/example_priv_key.pem`.
	require_client_cert	Boolean	Indicates whether a client certificate is required. Default: `NO`.
	server_name	String	Server name. Default: empty.
	srtp_secure_signaling	no yes sips	Indicates whether SRTP secure signaling is to be used
	timeout	Number	Timeout interval. Default: `0`.
	tls_enabled	Boolean	If set to YES, connection with TLS transport will be registered. Default: `NO`.
	use_srtp	disabled optional mandatory	Indicates whether to use SRTP
	verify_client	Boolean	Indicates whether clients must be verified. Default: `NO`.
	verify_server	Boolean	Indicates whether servers must be verified. Default: `NO`.

Specifying Behavior When A USB headset Is Not Available

The following behaviors can now be specified when a SIP Endpoint user does not have a working USB headset:

- Whether SIP Endpoint should automatically reject an incoming call

- The SIP error code to be sent to the inviting party

Support for this feature involves several configuration settings:

- endpoint:GSDefaultDevicePolicy:use_headset
- endpoint:GSDefaultSessionPolicy:reject_session_when_headset_na
- endpoint:GSDefaultSessionPolicy:sip_code_when_headset_na

Information about these settings is available in the table of SIP Endpoint Configuration Settings that appears elsewhere on this page.

You can tell whether SIP Endpoint has been instructed to use a USB headset by using the following method of `GSDevicePolicyDelegate`:

```
- (BOOL) useHeadset;
```

To determine whether SIP Endpoint will reject an incoming session when a USB headset is not available or to determine which SIP error code is sent if a USB headset is not available, use the following methods of `GSSessionPolicyDelegate`:

```
- (BOOL) rejectWhenHeadsetNa:(id<GSSession>) session;
- (NSString*) sipCodeWhenHeadsetNa:(id<GSSession>) session;
```

Configuring Message Waiting Indicator (MWI) Support

A Message Waiting Indicator (MWI) is usually an audio or visual signal that a voicemail or other type of message is waiting. SIP Endpoint SDK's MWI support involves several configuration settings:

- endpoint:basic:mailbox:user
- endpoint:basic:mailbox:server
- endpoint:basic:mailbox:transport
- endpoint:basic:mailbox:timeout

Information about these settings is available in the table of SIP Endpoint Configuration Settings that appears elsewhere on this page. You can use these settings to have SIP Server notify your application when new messages have been received by the subscribing mailbox. `GSMessageWaitingIndicationService` provides the following methods to control mailbox notification subscriptions:

```
-(GSResult) subscribeForMailbox:(GSMessageWaitingIndicationSubscription*) subscription;
-(GSResult) unsubscribeForMailbox:(GSMessageWaitingIndicationSubscription*) subscription;
```

Notifications are provided by `GSMessageWaitingIndicationNotificationDelegate`. Access to the MWI summary is provided by the following method:

```
- (void) state:(GSMessageWaitingIndicationState*)
         state forSubscription:(GSMessageWaitingIndicationSubscription*) subscription
```

These notifications encapsulate the following information:

```
subscription = theSubscription;
messagesWaiting = theMessagesWaiting;
messageSummary = theMessageSummary;
```

SIP Endpoint SDK OS X Sample Application

Warning

This documentation is outdated and is included for historical purposes only.

The easiest way to start using the SIP Endpoint SDK is with the bundled sample application. This application ships in the same folder as the SDK and is supplied as both a double-clickable application and as source code in the form of an Xcode project.

Running the Sample Application

Before running the sample application, you need to rebuild it in Xcode. To do that:

1. Open the `SipEndpoint Sample.xcodeproj` project file contained in the sample's `Src` folder.
2. Rebuild the project.

At that point, you can either run the application from within Xcode or you can double-click the application that is contained in the `Bin` folder.