SIP Endpoint SDK Developer's Guide

SIP Endpoint SDK 8.5.2NET

2/11/2022

Table of Contents

Developer's Guide

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
- 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 for .NET Product Page) to see the most recent news and updates about this product

Default SipEndpoint.config Settings

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:

Domain	Section	Setting
policy		
	endpoint	
		audio_qos
		include_os_version_in_user_agent_header
		include_sdk_version_in_user_agent_header
		ip_versions
		public_address
		refer_to_proxy
		rtp_inactivity_timeout
		rtp_port_min
		rtp_port_max
		signaling_qos
		sip_port_min
		sip_port_max
		sip_transaction_timeout
		video_max_bitrate
		video_qos
		vq_report_collector
		vq_report_publish
		webrtc_audio_layer
		answer_sdp_priority
		sip_port_binding
	session	
		agc_mode
		auto_accept_video

Domain	Section	Setting
		auto_answer
		auto_answer_delay
		dtmf_method
		echo_control
		noise_suppression
		dtx_mode
		reject_session_when_headset_na
		sip_code_when_headset_na
		vad_level
		ringing_enabled
		ringing_timeout
		ringing_file
		restart_audio_if_stuck
		reject_session_when_busy
		number_sessions_for_busy
		sip_code_when_busy
	device	
		audio_in_device For more information, see Audio Device Settings
		audio_out_device
		capture_device
		headset_name
		use_headset
codecs		

Domain	Section	Setting
— See Working with Codec Priorities		
proxies		
	proxy<*n*>	
		display_name
		domain
		password
		reg_interval
		reg_match_received_rport
		reg_timeout
		mailbox (sub-section of proxy<n>)
		password
		server
		timeout
		transport
		user
		nat (sub-section of proxy<n>)
		ice_enabled
		stun_server
		stun_server_port
		turn_password
		turn_relay_type
		turn_server
		turn_server_port
		turn_user_name

Domain	Section	Setting
system		
	diagnostics	
		enable_logging
		log_file
		log_level
		log_options_provider
		log_options_endpoint
		logger_type
		log_segment
		log_expire
		log_time_convert
		log_time_format
	security	
		cert_file
		tls_enabled
		use_srtp
	media	
		ringing_file

policy Domain

endpoint Section

audio_qos

Valid Values: Integer

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

Valid Values: `0`, `1`

Default Value: `1`

If set to 1, the user agent field includes the OS version the client is currently running on.

include_sdk_version_in_user_agent_header

Valid Values: `0`, `1`

Default Value: `1`

If set to 1, the user agent field includes the SDK version the client is currently running on.

ip_versions

Valid Values: `IPv4`, `IPv6`, `IPv4, IPv6`, `IPv6, IPv4`, or empty

Default Value: `IPv4, IPv6`

- IPv4—the application selects an available local IPv4 address; IPv6 addresses are ignored.
- IPv6—the application selects an available local IPv6 address; IPv4 addresses are ignored.

- IPv4,IPv6 or an empty—the application selects an IPv4 address if one exists. If not, an available IPv6 address is selected.
- IPv6,IPv4—the application selects an IPv6 address if one exists. If not, an available IPv4 address is selected.

Note: This parameter has no effect if the **public_address** option specifies an explicit IP address.

public_address

Valid Values: See description below

Default Value: Empty string which is fully equivalent to the `$auto` value

Local IP address or Fully Qualified Domain Name (FQDN) of the machine. This setting can be an explicit setting or a special value that the SDK uses to automatically obtain the public address.

Valid Values:

This setting may have one of the following explicit values:

- An IP address. For example, `192.168.16.123` for IPv4 or `FE80::0202:B3FF:FE1E:8329` for IPv6.
- A bare host name or fully qualified domain name (FQDN). For example, `epsipwin2` or `epsipwin2.us.example.com`.

This setting may have one of the following special values:

- $auto—The SDK selects the first valid IP address on the first network adapter that is active (status=up) and has the default gateway configured. IP family preference is specified by the policy.endpoint.ip_versions setting.
- $ipv4 or $ipv6—Same behavior as the $auto setting but the SDK restricts the address to a particular IP family.
- $host—The SDK retrieves the standard host name for the local computer using the gethostname system function.
- $fqdn—The SDK retrieves the fully qualified DNS name of the local computer. The SDK uses the GetComputerNameEx function with parameter ComputerNameDnsFullyQualified.
- An adapter name or part of an adapter name prefixed with $. For example, `$Local Area Connection 2` or `$Local`. The specified name must be different from the special values `$auto`, `$ipv4`, `$host`, and `$fqdn`.

If the value is an explicit host name, FQDN, or $fqdn, the Contact header includes the host name or FQDN for the recipient of SIP messages (SIP Server or SIP proxy) to resolve on their own. For all other cases, including $host, the resolved IP address is used for Contact. The value in SDP is always the IP address.

refer_to_proxy

Valid Values: `0`, `1`

Default Value: `0`

Specifies the destination of a referred INVITE.

- `0`—Send the INVITE to the URL specified in the Refer-To header of the REFER message.
- `1`—Send the INVITE to your configured SIP Proxy.

rtp_inactivity_timeout

Valid Values: `5-150`

Default Value: `150`

Suggested Value: `30`

Timeout interval in seconds for RTP inactivity.

rtp_port_min

Valid Values: `9000-65535`

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.

rtp_port_max

Valid Values: `9000-65535`

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

Valid Values: Integer

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

Valid Values: `1-65535`

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 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

Valid Values: `1-65535`

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.

sip_transaction_timeout

Valid Values: `1-32000`

Default Value: `4000`

SIP transaction timeout value in milliseconds. Valid values are 1 through 32000, with a default value of 4000. The recommended value is 4000.

video_max_bitrate

Valid Values: Integer

Integer value representing the maximum video bitrate.

video_qos

Valid Values: Integer

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.

webrtc_audio_layer

Valid Values: `0`, `1`, `2`, `1000`, `2000`, `3000`

Default Value: `0`

Specifies which audio layer is used for WebRTC.

- `0` — The audio layer is defined by the `GCTI_AUDIO_LAYER` environment variable — Core audio is used if this environment variable is not specified.
- `1` — Wave audio layer is used.
- `2` — Core audio layer is used.
- `1000` — Instructs the audio layer to open the microphone channel when the endpoint starts up, using the audio layer type defined by option `0`, and to keep it open until the endpoint is terminated.
- `2000` — Opens the speaker channel for the life of the endpoint, using the audio layer type defined by option `0`. Eliminates any delay in opening the audio device when an incoming or outgoing call is connected, for example in environments where audio device startup is slow due to a required restart of the Windows MMCSS service.
- `3000` — Opens the microphone and speaker channels for the life of the endpoint, using the audio layer type defined by option `0`.

Important

Keeping the audio channels permanently open eliminates any delay in connecting audio device to the call works around any issues with device occasionally not starting (or stopping) properly, at the cost of very small performance penalty.

answer_sdp_priority

Valid Values: `config`, `offer`

Default Value: `config`

- `config`—the endpoint selects the first codec from the codec configuration listed in both the codec configuration and the SDP offer.
- `offer`—the endpoint selects the first codec in the SDP offer listed in both the codec configuration and the SDP offer.

sip_port_binding

Valid Values: `0`, `1`

Default Value: `0`

- `0`—open the SIP port to listen on any interface.
- `1`—the SIP port binds to the interface specified by the public_address setting and listens only on this IP address.

session Section

agc_mode

Valid Values: `0`, `1`

Default Value: 1

If set to 0, AGC (Automatic Gain Control) is disabled; if set to 1, it is enabled. 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

Valid Values: `0`, `1`

This setting is only used in auto-answer scenarios when `auto_answer=1`.

If auto_accept_video is set to 1, both audio and video streams are accepted, otherwise incoming calls are answered as audio only, even if video is present in the offer.

auto_accept_video applies to a 3pcc answer when make-callrfc3275 is configured to 1 on the originating DN and a video codec is configured in the endpoint. auto_accept_video is not applied to a 3pcc answer when make-call-rfc3275 is configured to 2 on an originating DN, even if auto_accept_video is set to 1 and a video codec is configured in the endpoint.

auto_answer

Valid Values: `0`, `1`

If set to 1, all incoming calls should be answered automatically.

auto_answer_delay

Valid Values: Number in milliseconds

Default Value: `1500`

Time in milliseconds to wait before auto-answering. The recommended and default value is 1500 milliseconds.

dtmf_method

Valid Values: `Rfc2833`, `Info`, `InbandRtp`

Method to send DTMF.

echo_control

Valid Values: `0`, `1`

If set to 1, echo control is enabled.

noise_suppression

Valid Values: `0`, `1`

If set to 1, noise suppresion is enabled.

dtx_mode

Valid Values: `0`, `1`

If set to 1, DTX is activated.

reject_session_when_headset_na

Valid Values: `0`, `1`

If set to 1, the SDK should reject the incoming session if a USB headset is not available.

sip_code_when_headset_na

Valid Values: SIP Error Code

Defaul Value: `480`

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

Valid Values: `0-3`

Sets the degree of bandwidth reduction, from `0` for conventional VAD to 3 for aggressive high.

ringing_enabled

Valid Values: `0`, 1, 2, 3

Defaule Value: 1

Specifies whether to enable the ringing tone.

- 0—event Ringing disabled
- 1—event Ringing enabled
- 2—play ringtone internally (event Ringing disabled)
- 3—play ringtone internally and enable event Ringing.

ringing_timeout

Valid Values: Empty, `0`, or a positive number

Default Value: `0`

Specifies the duration, in seconds, of the ringing tone. If set to `0` or if the value is empty, the ringing time is unlimited.

ringing_file

Valid Values: Empty or the path to the ringing sound file. The path may be a file name in the current directory or the full path to the sound file.

Default Value: ringing.wav

Specifies the audio file that is played when the ringing tone is enabled with the ringing_enabled option.

Note that WebRTC does not support MP3 playback. The ringtone file for built-in ringing should be a RIFF (little-endian) WAVE file using one of the following formats:

- kWavFormatPcm = 1, PCM, each sample of size bytes_per_sample
- kWavFormatALaw = 6, 8-bit ITU-T G.711 A-law
- kWavFormatMuLaw = 7, 8-bit ITU-T G.711 mu-law

Uncompressed PCM audio must 16 bit mono or stereo and have a frequency of 8, 16, or 32 KHZ.

restart_audio_if_stuck

Valid Values: Empty, `0`, `1`

Default Value: `0`

- `0` or Empty—disable auto restart for stuck audio
- `1`—enable auto restart for stuck audio

reject_session_when_busy

Valid Values: Empty, `0`, `1`

Default Value: `0`

- `0` or Empty—disable rejection of a session when busy
- `1`—enable rejection of a session when busy

number_sessions_for_busy

Valid Values: Positive integer

Default Value: `1`

Sets the number of sessions before busy. Must be a positive integer.

sip_code_when_busy

Valid Values: Empty, 4xx, 5xx, 6xx

Default value: Empty

SIP error response code to use when busy. Can be set to any valid SIP error response code in the 4xx, 5xx, or 6xx range, for example, 486.

device Section

audio_in_device

Valid Values: String

Microphone device name.

For more information, see Audio Device Settings

audio_out_device

Valid Values: String

Speaker device name.

capture_device

Valid Values: String

Capture device name.

headset_name

Valid Values: String

The name of the headset model.

use_headset

Valid Values: `0`, `1`

If set to `0`, the audio devices specified in audio_in_device and audio_out_device are used by the SDK.

If set to 1, the SDK uses a headset as the preferred audio input and output device and the audio devices specified in audio_in_device and audio_out_device are ignored.

codecs Domain

See Working with Codec Priorities

proxies Domain

proxy<n> Section

display_name

Valid Values: String

Proxy display name.

domain

Valid Values:Any valid SIP domain

Defalut Value: Empty

A SIP domain is an application layer configuration defining the management domain of a SIP proxy. The configured value should include hostport and may include uri-parameters as defined by RFC 3261. The scheme, userinfo, and transport URI parameters are included automatically.

If set to an empty string, SIP Endpoint SDK for .NET uses the parameters from the Connectivity section to construct the SIP domain value as it did in previous versions.

password

Valid Values: String

Proxy password.

reg_interval

Valid Values: Integer

Default Value: `0`

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.

reg_match_received_rport

Valid Values: `0` or `1`

Default Value: `0`

This setting controls whether or not SIP Endpoint SDK should re-register itself when receiving a mismatched IP address in the received parameter of a REGISTER response. This helps resolve the case where SIP Endpoint SDK for .NET has multiple network interfaces and obtains the wrong local IP address. A value of `0` (default) disables this feature and a value of `1` enables re-registration.

reg_timeout

Valid Values: Number in seconds

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 is disabled, putting the endpoint in standalone mode.

mailbox Sub-section

Important

mailbox is a sub-section of the **proxy<n>** section.

password

Valid Values: String

Mailbox password.

server

Valid Values: String

Proxy server address and port for this mailbox.

timeout

Valid Values: Number in seconds

Default Value: `1800`

Subscription expiration timeout in seconds. If the setting is missing or set to `0`, the SDK uses a default timeout of `1800` seconds (30 minutes).

transport

Valid Values: `udp`, `tcp`, `tls`

Transport protocol to use when communicating with the server.

user

Valid Values: String

User ID for this mailbox.

nat Sub-section

> **Important**
> **nat** is a sub-section of the **proxy<n>** section.

ice_enabled

Valid Values: Boolean

Enable or disable ICE.

stun_server

Valid Values: String

STUN server address. An empty or null value indicates this feature is not used.

stun_server_port

Valid Values: Valid port number

Default Value: 3478

STUN server port value.

turn_password

Valid Values: String

Password for TURN authentication.

turn_relay_type

Valid Values: `0`, `udp`, `1`, or `tcp`

Type of TURN relay.

- `0` or `udp` for TURN over UDP.
- `1` or `tcp` for TURN over TCP.

turn_server

Valid Values: String

TURN server address. An empty or null value indicates this feature is not used.

turn_server_port

Valid Values:Valid port number

Default Value: 3478

TURN server port value.

turn_user_name

Valid Values: String

User ID for TURN authorization

`system` Domain

diagnostics Section

enable_logging

Valid Values: `0` or `1`

Disable or enable logging.

log_file

Valid Values: 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

Valid Values: Valid values for webrtc, `warning`, `state`, `api`, `debug`, `info`, `error`, `critical`

Example value: `gsip=2, webrtc=(error,critical)`

log_options_endpoint

Valid Values: `0`-4, same as **log_level**

Default Value: 2

Log levels: `0` = "Fatal"; 1 = "Error"; 2 = "Warning"; 3 = "Info"; 4 = "Debug"

5 = Logging disabled.

This setting should not be set higher than log_level setting.

logger_type

Valid Values: file

If set to `file` the log data will be printed to the file specified by the **log_file** value.

log_segment

Valid Values: `false`, number, or number in KB,MB, or hr

Default Value: `10 MB`

- false: No segmentation is allowed
- <number> or <number> KB: Size in kilobytes
- <number> MB: Size in megabytes
- <number> hr: Number of hours for segment to stay open

Specifies the segmentation limit for a log file. If the current log segment exceeds the size set by this option, the file is closed and a new one is created. This option is ignored if log output is not configured to be sent to a logfile.

log_expire

Valid Values: `false`, number, number `file`, number `day`

Deafult Value: `10` (store 10 log fragments and purge the rest)

- false: No expiration; all generated segments are stored.
- <number> or <number> file: Sets the maximum number of log files to store. Specify a number from 1—1000.
- <number> day: Sets the maximum number of days before log files are deleted. Specify a number from 1—100

Determines whether log files expire. If they do, sets the measurement for determining when they expire, along with the maximum number of files (segments) or days before the files are removed. This option is ignored if log output is not configured to be sent to a log file.

log_time_convert

Valid Values: `local`, `utc`

Default Value: `local`

- `local`: The time of log record generation is expressed as a local time, based on the time zone and any seasonal adjustments. Time zone information of the application's host computer is used.
- `utc`: The time of log record generation is expressed as Coordinated Universal Time (UTC).

Specifies the system in which an application calculates the log record time when generating a log file. The time is converted from the time in seconds since the Epoch (00:00:00 UTC, January 1, 1970).

log_time_format

Valid Values: `time`, `locale`, `ISO8601`

Default Value: time

- `time`: The time string is formatted according to the HH:MM:SS.sss (hours, minutes, seconds, and milliseconds) format
- `locale`: The time string is formatted according to the system's locale.
- `ISO8601`: The date in the time string is formatted according to the ISO 8601 format. Fractional seconds are given in milliseconds.

Specifies how to represent, in a log file, the time when an application generates log records. A log record’s time field in the ISO 8601 format looks like this: 2001-07-24T04:58:10.123.

security Section

cert_file

Valid Values: 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

Valid Values: `0`, `1`

Default Value: `0`

If set to 1, connection with TLS transport will be registered.

use_srtp

Valid Values: `optional`, `allowed`, `disabled`, `off`, `elective`, `both`, `enabled`, `force`, `mandatory`

Indicates whether to use SRTP:

- `optional` or `allowed`—do not send secure offers, but accept them
- `disabled` or `off`—do not send secure offers and reject incoming secure offers
- `elective` or `both`—send both secure and non-secure offers and accept either
- `enabled`—send secure offers, accept both secure and non-secure offers
- `force` or `mandatory`—send secure offers, reject incoming non-secure offers

media Section

ringing_file

Valid Values: Empty, String file name

Default Value: `rining.mp3`

The Ringing sound file name in the current directory or the full local path to the ringing sound file.

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

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

Starting with Windows Vista, Quality of Service (QoS) settings are supported on the level of the Windows operating system. This support allows you to configure policies to mark all of the packets sent by a particular application with the desired level of service.

Microsoft has documented this support at http://technet.microsoft.com/library/dd919203.aspx and at http://technet.microsoft.com/en-us/magazine/2007.02.cableguy.aspx.

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">
        <section name="diagnostics">
          <setting name="log_level" value="from 0 to 4"/>
      </domain>
```

Endpoint DNs

The following fields of DN objects can be configured under the SIP Server Switch:

- Number — Contains a numeric-only DN number. This number can be used to dial from a phone to reach this DN directly. The "@" character and domain names are not allowed. This number constitutes the username part of the endpoint's Address of Record (AOR).
- Type — Should be set to "Extension".
- Annex TServer/refer-enabled — Valid values are "true" or "false":
  - "true" indicates that REFER can be used to make a call or a transfer. (This value is supported by the SIP Endpoint SDK starting in release 8.0.000.11.)
  - "false" indicates that REFER is disabled for the above usage.
- Annex TServer/sip-cti-control — Valid values are "talk", "hold", "dtmf":
  - "talk" and "hold" are used to support the talk and hold BroadSoft extensions.
  - "dtmf" is used to support the sending of DTMF tones during third-party call control of an active call.
  - **Note:** As of SIP Endpoint SDK 8.1.2 for .NET, beep tone generation is no longer supported.

Audio Device Configuration

You can use SIP Endpoint SDK for .NET to configure audio devices such as headsets and microphones.

If you are using Windows 7, Windows 8, or Windows 2008 Server, this includes the ability to use devices with long device names.

The following settings can be used to control the behavior of these devices:

```
       <section name="device">
         <setting name="use_headset" value="1or0"/>
         <setting name="headset_name" value="Sennheiser USB"/>
         <setting name="audio_in_device" value="Sennheiser USB"/>
         <setting name="audio_out_device" value="Sennheiser USB"/>
       </section>

       <section name="session">
         <setting name="reject_session_when_headset_na" value="1or0"/>
         <setting name="sip_code_when_headset_na" value="480"/>
         <setting name="auto_answer" value="1or0"/>
   <setting name="auto_accept_video" value="1or0"/>
       </section>
```

The endpoint will automatically accept or reject a session, depending on whether or not a headset is available.

Configuration Notes

- If `use_headset` is set to `0` then the SDK will try to configure audio devices with the names that have been set in the `audio_in_device` and `audio_out_device` settings.
- If the configuration is not successfully completed, the default system devices will be selected.

Answer Actions and Dial Actions

The following tables describe answer actions and dial actions based on various combinations of settings.

Headset Availability
Use this table to determine what effect your headset settings will have on answer actions and dial actions. The last line indicates whether the action logic will consider the headset to be available or unavailable, or whether it will consider its state to be irrelevant.

Setting/State	Value		
use_headset	Yes	Yes	No
headset_name	Correctly specified	Incorrectly specified	Not applicable
Headset plugged in	Yes	Yes or No	Yes or No
Headset available	**Yes**	**No**	**Irrelevant**

Answer Actions
Depending on headset availability, as indicated in the previous table, the SDK provides the following options for answering an incoming session.

Setting/State	Value	
auto_answer	Yes	No

Setting/State	Value				
reject_session_when_headset_na		No	Yes	-	-
sip_code_when_headset_na	-	-	Code	-	-
Headset Available	Irrelevant	No	No	Yes	-
Answer	**Auto**	**Auto**	**Reject**	**Auto**	**Manual**

- If a headset is available, the session is answered as video or audio, depending on the media offer and the value of the `auto_accept_video` setting.
- If a headset is not available, the session is rejected with the SIP error code message specified in `sip_code_when_headset_na`.

Dial Actions
Outgoing sessions can be dialed using the headset availability criteria shown here, which are based on the inputs described in the Headset Availability table that appears above:

Setting/State	Value		
Headset Available	Yes	No	Irrelevant
Dial	**Dial**	**Error**	**Dial**

Defaults
If an endpoint is not configured in the configuration file, the following defaults will be used:

- `sip_code_when_headset_na=480`
- `auto_answer=0`
- `use_headset=0`

Ringer Device Configuration

Starting with SIP Endpoint SDK for .NET 8.5.200.57, you can configure ringer devices using the `ringer_device` and `ringing_enabled` settings:

```
<domain name="policy">
        <section name="device">
          <!-- Ringer -->
          <setting name="ringer_device" value="RingerDeviceName-HighPriority"/>
          <setting name="ringer_device" value="RingerDeviceName-LowPriority"/>
           . . .

        <section name="session">
          <!--
          settings "ringing_enabled" valid values:
          0 Ringing disabled
          1 event Ringing enabled (default value)
          2 built-in Ringing playback enabled
          3 built-in Ringing playback and event Ringing enabled
      NEW 4 built-in Ringing playback enabled using Ringer device
      NEW 5 built-in Ringing playback using Ringer device and event Ringing enabled
          -->
          <setting name="ringing_enabled" value="1"/>
           . . .
```

The SDK automatically chooses the ringer device according to configuration at startup and whenever the list of speaker devices changes. The procedure is the same as for audio devices. First, the SDK tries to find the device from the ringer_device setting. If this device is not found, the SDK uses the first system audio out device.

You can also configure the ringer device using the following methods for `interface IExtendedService`:

- `GsStatus SetRingerDevice(String^ name);`—use this method to programmatically set the ringer device.
- `String^ GetRingerName();`—returns the current ringer device name.

Advanced Settings and Configuration Options

The default configuration file does not contain all settings that may be used with the SIP Endpoint SDK; additional settings can be added to change certain behaviors. Some of these additional settings are discussed below.

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">
        <section name="device">
          <!-- Headset -->
          <setting name="use_headset" value="1"/>
          <setting name="headset_name" value="HeadsetName "/>
      </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">
        <section name="device">
          <!-- Headset -->
          <setting name="use_headset" value="0"/>

          <!-- Mic -->
          <setting name="audio_in_device" value="MicName"/>

          <!-- Speaker -->
          <setting name="audio_out_device" value="SpeakerName"/>
      </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.

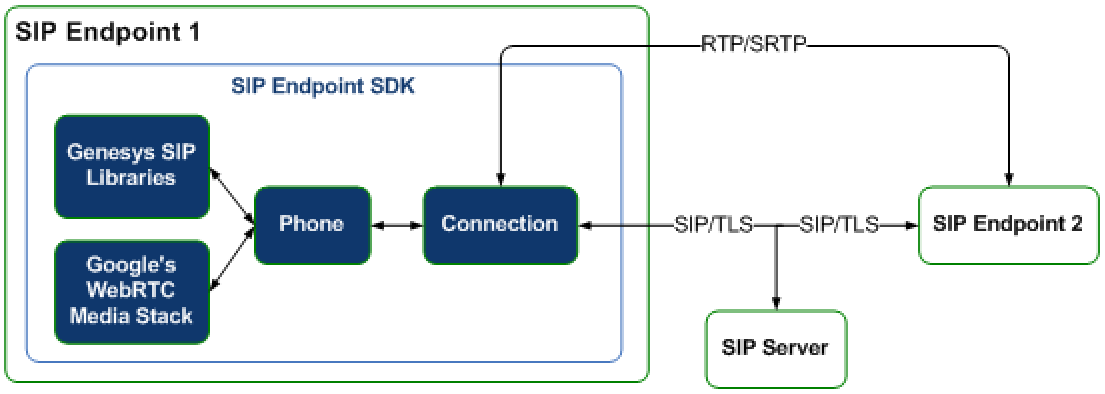


Figure 1: SIP Endpoint SDK Architecture with TLS and SRTP Enabled

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">
        <section name="security">
          <setting name="cert_file" value="ValueOfCertificateThumbprint"/>
          <setting name="use_srtp" value="enabled"/>
        </section>
      </domain>
```

Use the `use_srtp` setting to control media channel encoding. The setting `use_srtp` has the following valid values:

- `disabled`—SRTP is disabled (default)
- `optional` or `allowed`—SRTP is allowed
- `elective` or `both`—SRTP is allowed in both directions
- `force` or `mandatory`—SRTP is forced
- `enabled`—SRTP is enabled

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.

Important

The publish message is sent to the specified collector address only if the `vq_report_collector` parameter is configured with the `user@server:port;transport=udp` format. For example, `collector@127.0.0.1:5160;transport=udp`.

Settings:

```
<domain name="policy">
<section name="endpoint">
...
<!--
Valid values for Voice Quality (VQ) report publish setting (vq_report_publish):
0--VQ report is not published
1--VQ report is published to the collector at the end of the call--
  see the vq_report_collector setting information below
-->
<setting name="vq_report_publish" value="0"/>
<!--
Valid values for Voice Quality (VQ) report collector setting (vq_report_collector):
NULL or Empty--The VQ report is published to the proxy described in the
Connectivity section
FQDN or IP address along with port and transport--
 collector@SipServer.genesyslab.com:5060;transport=udp
-->
<setting name="vq_report_collector"
value="collector@SipServer.genesyslab.com:5060;transport=udp"/>
</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 for .NET release 8.1.2 and higher 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.

refer-enabled Setting

This is a SIP Server setting configured under the SIPS or DN Object. Set to `true` to enable SIP REFER.

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();
```

Working with Codec Priorities

Codecs are listed by name in the `codecs` domain of the configuration file. Codecs are listed by priority with codecs closer to the top of the list having higher priority. To disable a codec, comment out or remove the codec from the file.

The setting `payload_type` is an integer. For codec *h264* the valid values are between 96 and 127.

The setting `fmtp` is a string with valid values of RFC6184 for codec *h264* and RFC3555 for codec *g729*.

The codec *ulpfec/90000* supports the *vp8* codec with forward error correction.

Example

```
<domain name="codecs">
        <section name="PCMU/8000">
          <setting name="payload_type" value="0"/>
        </section>
        <section name="PCMA/8000">
          <setting name="payload_type" value="8"/>
        </section>
        <section name="G722/16000">
          <setting name="payload_type" value="9"/>
        </section>
        <section name="iLBC/8000">
          <setting name="payload_type" value="102"/>
        </section>
        <section name="iSAC/32000">
          <setting name="payload_type" value="104"/>
        </section>
        <section name="iSAC/16000">
          <setting name="payload_type" value="103"/>
        </section>
        <section name="vp8">
          <setting name="payload_type" value="100"/>
        </section>
        <section name="g729/8000">
          <setting name="payload_type" value="18"/>
          <setting name="fmtp" value="annexb=yes"/>
        </section>
        <section name="h264">
          <setting name="payload_type" value="108"/>
          <setting name="fmtp" value="profile-level-id=420028"/>
        </section>
        <section name="vp9">
           <setting name="payload_type" value="101"/>
        </section>
        <section name="ulpfec/90000">
          <setting name="payload_type" value="97"/>
        </section>
        <section name="opus/48000/2">
          <setting name="payload_type" value="120"/>
        </section>
      </domain>
```

Tip

For information on which codecs are supported by SIP Endpoint SDK for .NET, see the list of Supported Codecs in the SIP Endpoint SDK Overview.

Configuring Capture Devices

In order to define the list of preferred capture devices and their priorities, the devices should be added to the device policy section in the **SipEndpoint.config** file:

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

```
        <section name="device">
        <!--
          The priority of a device depends on its position in this section.
          The higher the position, the higher device priority.
          To disable a device it should be commented out or removed from the file.
        -->
. . .
          <!-- Capture -->
          <setting name="capture_device" value="CaptureDeviceName-HighPriority"/>
          <setting name="capture_device" value="CaptureDeviceName-LowPriority"/>
        </section>
      </domain>
```

API

The `IVideoControl` Interface has been extended with a method that allows you to set up a capture device based on:

- Its configuration settings in the **policy:device** section
- Its priority in that section, as described above

This method can be called at any time while SIP Endpoint SDK is active in order to carry out this task, as shown here:

```
        GsStatus SetCaptureDeviceFromConfig();
```

Application / SDK Interaction

The workflow between your app and the SIP Endpoint SDK is described in the following table:

App	SDK
Call: `ReleaseCaptureDevice()`	Releases currently used capture device
Call: `SetCaptureDeviceFromConfig()`	Finds matching capture device from the configuration file with the device from the array of all capture devices; if found, it sets this capture device as active
Call: `String^ GetCurrentCaptureDeviceName()`	Returns currently allocated capture device name
Compare with prioritized capture device name	

SIP Endpoint SDK Disaster Recovery and Geo-Redundancy

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.

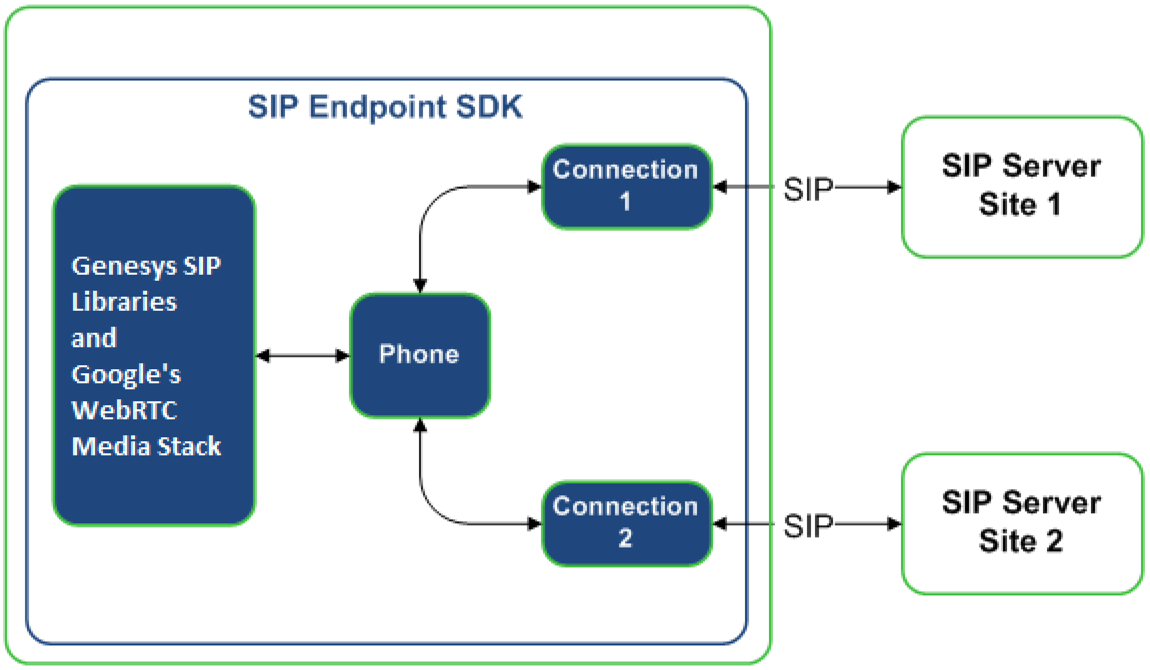


Figure 1: Geographical Redundancy and Disaster Recovery Architecture.

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.

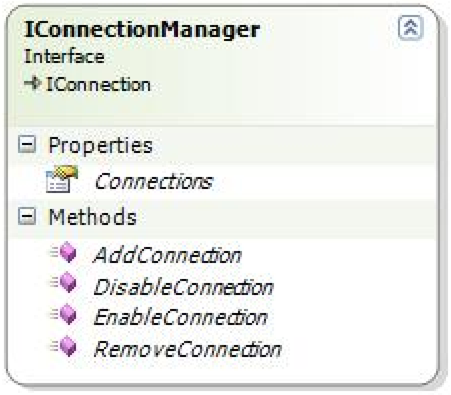
Working with Multiple Connections

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

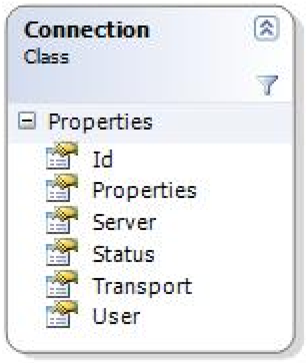
Handling Multiple Connections



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:

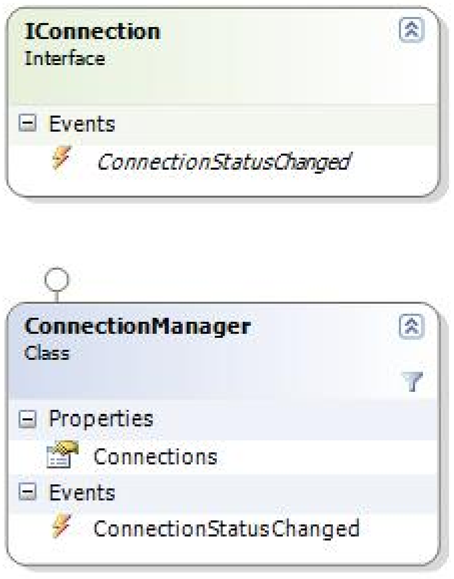


Managing Connection Status

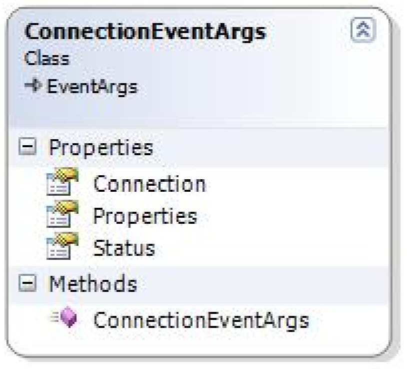
Notification of changes to the connection status is given by the ConnectionStatusChanged event.

```
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="reg_interval" value="10"/>
    </section>
    <section name="proxy1">
        ...
        <setting name="reg_interval" value="20"/>
    </section>
</domain>
```

SIP Endpoint SDK .NET QuickStart Application

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>\QuickStart\QuickStart2013\Src` folder or the `<SIP Endpoint SDK Folder>\QuickStartExe` 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 ="dn0" server="SipServer0.domain.invalid:5060" protocol="udp"/>
    <Connectivity user ="dn1" server="SipServer1.domain.invalid: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 QuickStartVS2013.exe.

Or you can build and run the Visual Studio project:

1. Open the <SIP Endpoint SDK Folder>\QuickStart\QuickStart folder.
2. Double-click QuickStart2013.sln.
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();
```

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.

Audio Device Settings

SIP Endpoint SDK for .NET uses a complex set of criteria for determining how to select its audio input and output devices. The following sections describe:

- The basic priority settings for audio input and output devices
- The rules that are used to select an audio device, auto-answer a call, and reject a call
- The ways in which various combinations of settings affect audio device selection, auto-answer, and call rejection

Basic Priority

Headsets and other audio input devices are configured by using the following parameters:

- `headset_name`
- `audio_in_device`
- `audio_out_device`

You can set the priority for these devices by changing the order of their entries in the configuration file. If none of the audio devices that are accessible to the endpoint match the device names in the configuration file, the SDK will pick up the first available devices from the WebRTC lists for audio devices.

Examples

In the following example, the Plantronics C320 has a higher priority than the Plantronics D100, since it appears first in the file:

```
<setting name="headset_name" value="Plantronics C320"/>
<setting name="headset_name" value="Plantronics D100"/>
```

Selection Rules

The following rules are used to select an audio device, auto-answer a call, and reject a call.

Audio Device Selection

The procedure for audio device selection is applied on startup and every time any changes are made to device presence (such as when a new device is plugged in or an existing device is removed):

1. The first device in the applicable list that is present in the system is selected when possible. This device

(or devices) will either be specified by `headset_name` or by `audio_in_device` and `audio_out_device`, depending on whether `use_headset` has been enabled.

2. If none of the configured devices are present (or if the configuration list is empty), then the SDK will select the audio devices using the priority that has been provided by WebRTC, based on the order of the available devices in its device list.

Auto-Answer

In cases where either of the following conditions is met, the auto-answer functionality is blocked (a policy of `should answer` returns unknown, although a manual answer is still possible):

- `use_headset` is set to 1, and none of the devices listed in the `headset_name` settings is currently present (but session rejection is not applicable, that is, `reject_session_when_headset_na` has been set to 0)
- The SDK was unable to find any usable microphone or speaker device (applicable to cases when `use_headset` is set to 0)

Finally, when `auto_answer` is set to 1 and the auto-answer functionality is not blocked (and the call was not already rejected), the SDK answers the incoming call automatically (the `should answer` policy returns `true`). If `auto_accept_video` is set to 1, then both audio and video streams are accepted, otherwise the call is answered as audio-only, even if video is present in the offer.

Rejecting A Call

For backward compatibility with previous releases, a call can only be rejected when both of the following conditions are met (a policy of `should answer` returns `false`):

- Both `use_headset` and `reject_session_when_headset_na` are set to 1
- None of the devices listed in the `headset_name` settings is currently present

When these conditions are met, an incoming call is rejected with a SIP response code as configured in the `sip_code_when_headset_na` setting. If the setting is missing or the value does not belong to the valid range of 400 to 699, then the default of 480 (`Temporarily Unavailable`) is used.

In addition, when these conditions are met, the SDK will refuse to initiate any new calls, that is, it will reject outgoing calls.

Note that the availability of a fallback device (selected by Step 2 in the Audio Device Selection section) does not affect call rejection.

Combinations of Settings

The following combinations of settings affect audio device selection, auto-answer, and call rejection in the ways described below.

use_headset=1

Headset is Available The SDK considers a headset to be available if a headset was found by name in the list of `headset_name` entries. (The highest priority device in the list is selected). Outgoing calls can be initiated.	`auto_answer=1`	Incoming calls are answered automatically: • As audio if `auto_accept_video=0` • As audio with video if the call has video and `auto_accept_video=1`
	`auto_answer=0`	Incoming calls are answered manually and the user explicitly selects whether or not video streams should be accepted (using the `has_video` parameter supplied in the `gs_session_info` argument)
Headset is Not Available The SDK decides that no headset is available if a headset was not found by name in the list of `headset_name` entries. An audio device is still assigned, if possible (that is, if any supported devices are present in the system), using the first available audio input and output devices from the list compiled by WebRTC.	No auto-answer is possible in this sub-case, so the `auto_answer` setting is not used	`reject_session_when_headset_na=1` • Incoming calls are automatically rejected • Outgoing calls are blocked
		`reject_session_when_headset_na=0` • Incoming calls can be answered manually—it is assumed that the agent will plug the headset in (or use an available non-headset device, if applicable) before answering the call • Outgoing calls can be initiated—it is the agent's responsibility to ensure that the appropriate audio devices are available before the call is answered by the remote side

use_headset=0

Audio devices are configured using the names from the `audio_in_device` and `audio_out_device` settings. The SDK selects the highest-priority input and output devices from that list or, if no valid devices are found in that list, from the first available devices in the list compiled by WebRTC. Outgoing calls can be initiated.

Both microphone and speaker are available	`auto_answer=1`	Incoming calls are answered automatically:

		• As audio if `auto_accept_video=0` • As audio with video if the call has video and `auto_accept_video=1`
	`auto_answer=0`	Incoming calls are answered manually and the user explicitly selects whether or not video streams should be accepted (using the `has_video` parameter supplied in the `gs_session_info` argument)
Either microphone or speaker is not available • Incoming calls can be answered manually—it is assumed that the agent will plug in the headset (or use an available non-headset device, if applicable) before answering the call • Outgoing calls can be initiated—it is the agent's responsibility to ensure that the appropriate audio devices are available before the call is answered by the remote side	No auto-answer is possible in this sub-case, so the `auto_answer` setting is not used	Auto-rejection is not applicable, so the `reject_session_when_headset_na` setting is not used

SIP Endpoint SDK Video Support

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

This article also outlines the requirements for processing High Definition video.

High Definition Video Requirements

Important

If you plan on supporting High Definition (HD) video, please take the following requirements into account.

Due to the complex processing required for High Definition video, all of the endpoints that are involved in a video conversation must run on computers that meet a certain minimum hardware performance level. As the actual CPU performance can no longer be accurately measured in MHz and since video processing performance depends on a wide variety of factors, Genesys recommends that you use the free benchmarking tool NovaBench to assess whether your hardware meets the requirements for successful HD video processing.

For 720p HD video, the **minimum** requirements are (in addition to what the camera manufacturer requires):

- A total NovaBench score of at least 500 (with a Graphic Test sub-score of at least 12), with recommended scores of at least 800 and 20, respectively
- 2 GB RAM
- 1 Mbps upload and download speed (for a total bandwidth of 2 Mpbs)

SIP Endpoint SDK for .NET does not currently support video resolutions higher than 720p.

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

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 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

Method	Description
	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);
```

Adding Video to an Audio-Only Call

In order to add video to an existing session, one of the supported video codecs should be specified in the **SipEndpoint.config** file:

```
      <domain name="codecs">
```

```
. . .
        <section name="vp8">
          <setting name="payload_type" value="100"/>
        </section>
        <section name="h264">
          <setting name="payload_type" value="108"/>
          <setting name="fmtp" value="profile-level-id=420028"/>
        </section>
        <section name="vp9">
           <setting name="payload_type" value="101"/>
        </section>
      </domain>
```

If **auto_accept_video** is enabled and the other party adds video during an active audio call, video is automatically added to the endpoint. However, if **auto_accept_video** is not enabled and the other party adds video during an active audio call, then the user is prompted to either accept or reject the video.

API

The `ICallControl` Interface has been extended with these methods:

```
        GsStatus SendVideoOffer(int sessionId);
        GsStatus AnswerVideoOffer(int sessionId, bool accept);
        GsStatus RemoveVideoStream(int sessionId);
```

You can add video to audio-only calls using both built-in and external video frame renderers.

Application Notes

Local video should be started before you add the video. If the media offer is accepted at this point, the video becomes bidirectional. If there is a request to remove the video, the local video is disconnected from the session.

Audio-video scenarios

Video Offer Rejected By Remote Party

Local party	Remote party	API
Establish audio-only session		
Voice to video escalation		
Start local video		
Request to add video		`SendVideoOffer(sessionId)`
	Receive session status	`SessionStatus.MediaOffer`
	Reject video offer	`AnswerVideoOffer(sessionId, FALSE)`
Receive session status		SessionStatus.MediaRejected
Audio-only session established		

Video Offer Accepted By Remote Party

Local party	Remote party	API
Add video to audio-only call		
Voice to video escalation		
Start local video		`StartLocalVideo()`
Request to add video		`SendVideoOffer(sessionId)`
	Receive session status	`SessionStatus.MediaOffer`
	Accept video offer	`AnswerVideoOffer(sessionId, TRUE)`
Receive session status		`SessionStatus.MediaAccepted`
Show remote video	Show remote video	`StartRemoteVideo(sesionId)`
Bidirectional video session established		

Video Removed From The Call

Local party	Remote party	API
Existing bidirectional video session		
Remove video		`RemoveVideoStream(sessionId)`
Receive session status	Receive session status	`SessionStatus.MediaRejected`
Close remote video	Close remote video	`StopRemoteVideo(sesionId)`
Voice only session		

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

NAT Traversal

SIP Endpoint SDK supports NAT traversal for restrictive firewalls and routers in the following general scenarios:

- Enterprise to Cloud
- Consumer to Cloud
- Home Agent to Cloud
- Mobile to Enterprise
- Mobile to Cloud

To configure NAT, see the `nat` section of the Default Configuration Settings.

Important

- Double NAT is not supported.
- NAT translation will not occur with a NAT server that does not support UDP fragmentation. An example of such a server is Microsoft RRAS (Routing and Remote Access Service).
- NAT functionality was tested with the following NAT and STUN/TURN servers:
  - NAT Server—Windows 2008 NAT Server using RRAS service
  - STUN/TURN Server—https://code.google.com/p/rfc5766-turn-server/
- When using UDP transport for an endpoint behind the NAT, you must set the **reg_timeout** option to a value not exceeding twice the binding timeout for your particular NAT implementation.

  Periodic REGISTER messages serve as a keep-alive mechanism necessary to keep the NAT channel open. Genesys recommends a value of `60` for **reg_timeout** (REGISTER sent every 30 seconds) which should work with most NAT implementations.

Re-registration

Use the `reg_match_received_rport` setting in the proxyN section to control re-registration in cases where the received/`rport` values in the REGISTER response do not match local values. A value of 0 (default) disables this feature and a value of 1 enables re-registration.

```
<domain name="proxies">
    <section name="proxyN">
      <setting name="reg_match_received_rport" value="0 or 1"/>
. . .
    </section>
. . .
```

```
</domain>
```

Important

For re-registration to work correctly, the registrar must support RFC 3581. Genesys SIP Server **does not** support RFC 3581. In cases where re-registration may interfere with NAT traversal, the `reg_match_received_rport` setting can be used to turn off re-registration.