SIP Endpoint SDK Developer's Guide

Support for Dynamic Reconfiguration

12/13/2025

Contents

Support for Dynamic Reconfiguration

> **Important**
> This feature was introduced as part of the 9.0.006.10 release and updated in 9.0.016.03 release.

This feature provides additional options to support regex matching and specify codec priorities.

This page describes the API and rules for dynamic reconfiguration of SIP Endpoint SDK from an application. Implementation provides two methods: one to get access to current configuration and another to change that configuration dynamically.

OSX

The following methods are defined in the GSEndpoint protocol:

```
@protocol GSEndpoint <NSObject>
/**
 Get endpoint configuration setting
 @param key
 @returns string value of the requested key string
 @returns empty string if key-value pair does not exists
 @returns empty string if key null or empty
 */
- (NSString*) getConfigSettingForKey:(NSString*) key;
/**
 Set endpoint configuration setting
 @param value
 @param key
 @returns result of the operation
 */
- (GSResult) setConfigSettingValue:(NSString*) value forKey:(NSString*) key;
```

.NET

The following methods are defined in the IExtendedService interface:

```
public interface class IExtendedService
{
  void SetConfigStringSetting(String^ key, String^ value);
  String^ GetConfigStringSetting(String^ key);
}
```

Detailed Description

The "key" value (for both configuration methods) should be in either one of these two forms:

1. for reference to Connectivity parameters in "Basic" container -- "*name* `:`*N*", where
   - *name* is the attribute name, one of `user`, `server`, `protocol`, `transport` (synonym for `protocol`).
   - *N* refers to Connectivity line index, starting from 0, (i.e., the setting "`user:0`" refers to the *user* parameter in the first Connectivity line, and "`user:1`" refers to the second line.)
2. for reference to all setting in Genesys container -- *domain*`.`*section*`.[`*subsection*`.]`*setting*, where
   - *domain* is the XML domain element and must be one of `policy`, `codecs`, `proxies` or `system`
   - *section* and `setting` refer to corresponding XML schema elements
   - *subsection* is optional section name, as used currently for NAT options

> **Important**
>
> Historically, SDK for .NET and OS X used different delimiters in option keys. For backward compatibility, in the current SDK version, dot and colon in the key value may be used interchangeably.

When do the Changes Take Effect?

While any configuration setting may be changed any time, not all changes take effect immediately. Particularly, for most cases:

- Connectivity parameters, settings in `proxies` domain, and `system.security` section take effect when connection is activated,
- Settings in `policy.session` section and in `codecs` domain take effect for the next session created,
- Settings in `policy.device` section take effect next time device is going to be selected.

The following settings in policy.endpoint section that take effect for the next session (without Endpoint restart):

- `include_os_version_in_user_agent_header`
- `include_sdk_version_in_user_agent_header`
- `answer_sdp_priority`
- `defer_device_release`
- `refer_to_proxy`
- `vq_report_publish`
- `vq_alarm_threshold`

The following settings in policy.endpoint section take effect only when connection is activated:

- `public_address`
- `sip_port_binding`

The following settings in policy.endpoint section do not take effect without full SDK or application restart:

- ip_versions
- `sip_port_min`, `sip_port_max`
- `rtp_port_min`, `rtp_port_max`
- `tcp_port_min`, `tcp_port_max`
- `rtp_inactivity_timeout`
- `sip_transaction_timeout`

Code Samples

OSX

The OSX code samples are written in assumption that application code uses class with 'ep' property referring to the GSEndpoint object and it includes the following in the app header:

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

and that property is initialized in the app implementation:

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

Changing Session Policy Auto-answer

This example shows how to get the current value and set the policy, `auto-answer` value to *1* (true). Note that policy change will not affect any calls that are already ringing on the endpoint but will take effect from the next call onwards.

```
NSString *oldAA = [self.ep getConfigSettingForKey:"policy.session.auto_answer"];
GSResult result = [self.ep setConfigSettingValue:"1" forKey:"policy.session.auto_answer"];
if (result == GSResultOK)
     NSLog(@"Auto-answer changed from %@ to 1.",oldAA);
else NSLog(@"Error %d changing Auto-answer from %@.", result, oldAA);
```

A similar approach can be used for other settings that do not require the connection or SDK to be restarted to be take effect.

Changing Connectivity Parameters

The following sample shows how to reconnect to a different SIP Server location. To do so, the application should:

- disable a connection with certain `configId` or `connectionId`
- change the server location
- enable the connection again

1. Obtain a connection reference by:
   - `configId` (the position in connections list of the current configuration, starting from *0* index)

     ```
     GSSipConnection *connection = [[self.ep connectionManager]
      connectionByConfigId:configId];
     ```

   - `connectionId` (an ID that is set in the `connectionId` property when enabling the connection)

     ```
     GSSipConnection *connection =
      [[self. ep connectionManager] connectionByConnectionId:connectionId] ;
     ```

2. Changing the server location to `new_server` for given connection:

```
[connection disable]; // that operation cannot fail (always returns GSResultOK)
NSString *key = [NSString stringWithFormat:@"server:%d", connection.configId];
GSResult result = [self.ep setConfigSettingValue:new_server forKey:key];
if (result == GSResultOK) result = [connection enable];
if (result == GSResultOK)
     NSLog(@"Connection changed to %@.", new_server);
else NSLog(@"Error %d changing connection.", result);
```

Changing Endpoint Setting with Full SDK Restart

The following sample shows how to change a global value (e.g., `policy.endpoint.sip_transaction_timeout`) and then perform a full SDK restart.

```
[self.ep stop] // stop endpoint
  [self.ep setConfigSettingValue:@"5000" forKey:@"policy.endpoint.sip_transaction_timeout"];
if ([self.ep configure]) { // re-start endpoint
  self.ep.notificationDelegate = self;
  self.ep.connectionManager.notificationDelegate = self;
  self.ep.sessionManager.notificationDelegate = self;
  self.ep.deviceManager.notificationDelegate = self;
  [self.ep activate];
  [self.ep.deviceManager configure];
  self.connections = [self.ep.connectionManager allConnections];
}
else NSLog(@"Error restarting");
```

.NET

Code samples in this section are written in assumption that application code uses class with `endpoint` property.

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

and the following initialization code:

```
this.endpoint = SipEndpoint.EndpointFactory.CreateSipEndpoint();
this.extendedService = this.endpoint.Resolve("IExtendedService") as ExtendedService;
this.connectionManager = this.endpoint.Resolve("IConnectionManager") as ConnectionManager;
```

Changing Session Policy Auto-answer

This example shows how to get the current value and set the policy, `auto-answer` value to *1* (true).

> **Important**
>
> Any change will not affect any calls that are already ringing on the endpoint but will take effect from the next call onwards.

```
String oldAA= this.extendedService.GetConfigStringSetting("policy.session.auto_answer");
GsStatus result = this.extendedService.SetConfigStringSetting("policy.session.auto_answer",
"1");
if (result == GsStatusSuccess)
     this.extendedService.LogPrint(4,"Auto-answer changed from " + oldAA +" to " + newAA);
else this.extendedService.LogPrint(4,"Error" + result + "changing Auto-answer to " + newAA);
```

A similar approach can be used for other settings that do not require the connection or SDK to be restarted to be take effect.

Changing Connectivity Parameters

The following sample shows how to reconnect to a different SIP Server location. To do so, the application should:

- disable a connection with certain `configId` or `connectionId`
- change the server location
- enable the connection again

These steps can be performed as following:

```
this.connectionManager.DisableConnection(connectionId); // that operation cannot fail (always
returns GsStatusSuccess)
String key = "server:" + connection.configId.ToString();
this.extendedService.SetConfigStringSetting(key, new_server);
this.connectionManager.EnableConnection(connectionConfigId);
```

If the application requires the `connectionId` value for a known `connectionConfId`, the following method may be used:

```
 public int GetConnectionIdByConfigId(int connectionConfigId)
 {
   foreach (Connection conn in this.connectionManager.Connections)
   if (conn.ConfId == connectionConfigId) return conn.Id;
```

```
   return -1; // negative result means "not found"
 }
```

Changing Endpoint Setting with Full SDK Restart

The following sample shows how to change a global value (e.g., `policy.endpoint.sip_transaction_timeout`) and then perform a full SDK restart.

```
this.extendedService.StopCore(); // stop endpoint
this.extendedService.SetConfigStringSetting("policy.endpoint.sip_transaction_timeout",
"5000");
this.extendedService.RestartCore();
```