

# ADOBE® FLASH® MEDIA GATEWAY 2.0

## LEG SERVICE API REFERENCE

© 2010 Adobe Systems Incorporated. All rights reserved.

Adobe® Flash® Media Gateway Leg Service API Reference for Windows®.

This API reference is licensed for use under the terms of the Creative Commons Attribution Non-Commercial 3.0 License. This License allows users to copy, distribute, and transmit the reference for noncommercial purposes only so long as (1) proper attribution to Adobe is given as the owner of the reference; and (2) any reuse or distribution of the reference contains a notice that use of the reference is governed by these terms. The best way to provide notice is to include the following link. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/>.

Adobe, the Adobe logo, Adobe AIR, AIR, Flash, and Flash Lite are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Intel is a registered trademark of Intel Corporation in the U.S. and other countries. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are the property of their respective owners. All other trademarks are the property of their respective owners.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

# Contents

<b>Acronyms</b>	1
<b>Chapter 1: Introduction</b>	
Call leg	2
Call leg service	2
Call leg states	3
<b>Chapter 2: Leg service APIs</b>	
Method summary	4
Event handler summary	4
<b>Chapter 3: SSAS methods</b>	
CallLegService.setLogLevel(level)	5
CallLegService.setServer(client)	5
CallLegService.createCall(callerAddress, destAddress, [callerIdName],[callerIdNumber] ,[callLegOptions])	6
CallLegService.setLegStateRing(legID)	6
CallLegService.setLegStateSendReceive(legID)	7
CallLegService.setLegStateHangup(legID, cause)	7
CallLegService.pushNewStream(legID, streamName)	7
CallLegService.removeStream(legID, streamName)	8
CallLegService.sendLegMessage(legID, info)	8
CallLegService.setWriteCodec(legID, codecName)	9
CallLegService.setSilenceLevel(legID, iVal)	9
CallLegService.setWriteAudioGain(legID, iVal)	9
CallLegService.setServiceID(strServiceID)	10
CallLegService.getServiceID()	10
<b>Chapter 4: Event handlers</b>	
CallLegService.onStatus(info)	11
CallLegService.onLeg(info)	11
CallLegService.onLegStatus(info)	12
CallLegService.onLegMessage(info)	13
CallLegService.onError(info)	13
<b>Chapter 5: Sample usage</b>	
Overview	15
Creating a telephony application using FMG leg service API	15
Best practices	19

# Acronyms

FMG	Flash Media Gateway
SIP	Session Initiation Protocol
RTP	Real Time Protocol
RTMP	Real Time Messaging Protocol
DTMF	Dual-tone Multi-Frequency
FMS	Flash Media Server
UAC	User Agent Client
UAS	User Agent Server

# Chapter 1: Introduction

Leg service is a set of programmable APIs that can be utilized by a server-side application on Flash® Media Server (FMS). These Leg service APIs utilize the functionalities and workflows provided by Flash® Media Gateway (FMG). Thus, FMG acts as a RTMP client connecting to FMS.

A set of handlers is required for an FMS server-side application to use FMG. Leg service provides a set of methods and the handlers required to invoke local methods from FMG. Leg service also provides an interface to invoke remote methods at FMG.

Leg service APIs enable an FMS server-side application to create/receive and manage multiple VoIP calls with SIP endpoints, which are reachable through FMG. FMG provides an interface to configure the XML files required to set up the interaction between endpoints such as:

- RTMP to RTMP
- RTMP to SIP
- SIP to SIP
- RTMP to Playfile
- SIP to Playfile
- FMS server-side application to SIP

A typical server-side application on FMS extends these VoIP capabilities to the RTMP clients (including Adobe® Flash® Player) in a customized mode. This customization enhances flexibility, control, and creativity.

## Call leg

A call leg is the data structure representing an active connection to one logical endpoint, such as a Flash phone. Each call leg is identified by a UUID string (leg ID), which is shared between FMG and FMS applications

## Call leg service

Call leg service is a SSAS telephony API class that extends the SSAS client connection from FMG leg service and uses it to communicate events/requests. SSAS application needs to explicitly identify connect/disconnect from FMG leg service client and pass on the information to leg service object using the setServerAPI. Leg service API does not keep binding of actual flash clients with call leg ID. It is up to the SSAS application to maintain an appropriate association based upon usage. Call leg service object at SSAS is limited to maintaining internal state and association of call legs with streams. The scope is limited to call legs ending at the current FMS application instance.

FMG connects to an FMS application if SSAS API services.request("<service-name>") is invoked with the appropriate <service-name>. See rtmp.xml configuration in *Adobe® Flash® Media Gateway Installation and Configuration Guide* for more information.

Call leg service provides APIs for RTMP call leg-specific initialization (acceptance, RTMP stream negotiation), termination, and state updates. Upon initiating, a new call leg is created on FMG. The call leg goes out of scope after conversation is finished (at call hangup). For each call leg, a stream is received with info object of onLeg handler, which FMG publishes to FMS. This stream should be subscribed to receive audio from FMG. FMS-to-FMG stream names should be explicitly specified using pushNewStream or removeStream APIs. Call leg service provides APIs to interact remotely with a call leg structure on FMG; for example, sending/receiving DTMF digits.

## Call leg states

The state of a call leg is maintained by FMG and the server side application is notified about each state change by invoking the onLegStatus event handler. While initiating a state change, SSAS API has limited permissions to make state transitions. Some of the permitted state transactions are between the following leg states:

- init to ring
- init to sendrecv
- ring to sendrecv
- [any active state] to hangup
- [any active state] to hold
- leg.state.hold to sendrecv
- leg.state.hold to hangup

**Note:** The active states for a call leg are *leg.state.ring*, *leg.state.sendrecv*, *leg.state.init*, and *leg.state.execute*.

Permissions can also depend on the type (outgoing/incoming) of call and its present state. Normal conversation can be expected during *leg.state.sendrecv* state. See [Sample usage](#) for more details.

# Chapter 2: Leg service APIs

## Method summary

API method	Description
CallLegService.setLogLevel	Sets the logging level in the leg service library.
CallLegService.setServer	Initializes leg service by providing the client object of FMG Leg Service connection or stops leg service by providing NULL.
CallLegService.createCall	Creates a new RTMP call leg on FMG.
CallLegService.setLegStateRing	Sets the state of an existing call leg to leg.state.ring, to indicate ringing.
CallLegService.setLegStateSendRecv	Sets the state of a call leg to leg.state.sendrecv, to indicate that call media is set up and streams have been subscribed/published.
CallLegService.setLegStateHangup	Sets the state of an active call leg to leg.state.hangup to end the conversation.
CallLegService.pushNewStream	Indicates to FMG to subscribe a streamName published by a flash phone for a call leg.
CallLegService.removeStream	Indicates to FMG to no longer use a particular streamName for flash phone to FMG media.
CallLegService.sendLegMessage	Sends DTMF digits.
CallLegService.setWriteCodec	Sets the codec of FMG to FMS stream for a call leg.
CallLegService.setSilenceLevel	Sets the <code>silenceLevel</code> property of an RTMP call leg.
CallLegService.setSilenceTimeout	Sets the <code>silenceTimeout</code> property of an RTMP call leg.
CallLegService.setWriteAudioGain	Increase or decrease the audio gain of incoming voice on a callLeg.
CallLegService.setServiceID	Replaces the default serviceID custom serviceID for the CalllegService Object.
CallLegService.getServiceID	Get ServiceID to uniquely identify a CallLegService Object

## Event handler summary

Event handler	Description
CallLegService.onStatus	Invoked whenever the state of leg service changes at FMG.
CallLegService.onLeg	Invoked each time a new RTMP call leg is created for either an incoming or outgoing call.
CallLegService.onLegStatus	Invoked each time any RTMP call leg changes its state.
CallLegService.onLegMessage	Invoked each time an RTMP call leg receives a DTMF digit or indication.
CallLegService.onError	Invoked in case critical error occurs in FMG for example when a request to create call fails.

# Chapter 3: SSAS methods

## CallLegService.setLogLevel(level)

You can use this API to modify the selected log level of leg service APIs. You can view all logs in the FMS application.log file corresponding to the server-side application.

### Input

The input parameter, `level`, can have the following values:

- `error`: Logs only error messages.
- `warning`: Logs warnings and error messages.

**Note:** *warning is the default log level.*

- `info`: Logs informational messages. These logs include messages that are generated when an API or handler is invoked.
- `verbose`: Logs each API or handler invocation, along with all the function parameters.

### Result

The API returns “true” when log level has been set successfully; otherwise, it returns “false”.

## CallLegService.setServer(client)

You can use this API to:

- initialize leg service by providing the client object of FMG leg service connection, or
- stop leg service by providing NULL

### Input

The input parameter, `client`, can have the following values:

- FMG call leg service connection: To initialize a leg service.
- NULL: To stop a running call leg service. `setServer(NULL)` sends the hangup status to all active call legs on this service and closes it.

**Note:** *If a service is already running, repeated calls to this API with the same "client" will not reset the service.*

### Result

This API doesn't return any value.



## CallLegService.createCall(callerAddress, destAddress, [callerIdName],[callerIdNumber] ,[callLegOptions])

You can use this API to create a new call leg on FMG.

### Input

The input parameters for this API are as follows:

- **callerAddress:** The Flash phone ID with which the SSAS application recognizes the phone. It should be a string that contains digit(s). It can be NULL.
- **destAddress:** The number that indicates the default dial-plan setup in workflow.xml. The dial plan uses this address to identify the number of the actual recipient of the call. For example, <9><international phone number>.

**Note:** *destAddress is a mandatory field and can't have its value set to NULL.*

- **callerIdName:** A name for the caller ID, which can be displayed at the remote end of an outgoing call. This is an optional field.
- **callerIdNumber:** A number for the caller ID, which can be displayed at the remote- end of an outgoing call. This is an optional field.
- **callLegOptions:** An object of class CallLegOptions. This object contains methods to set certain call specific parameters e.g.for requesting privacy in an outgoing call method setPrivacy may be used. Usage: callOptions = new CallLegOptions(); callOptions.setPrivacy(true);

### Result

**On success:** A reference ID string (refID), when the request has been successfully forwarded to FMG.

The application might store the `refID` field to identify/match the corresponding new leg confirmation on receiving the onLeg handler from FMG.

**On failure:** NULL.

## CallLegService.setLegStateRing(legID)

You can use this API to request FMG to change the state of a call leg to Ringing.

### Input

The input parameter is `legID`. This is an optional transition, which can be requested to denote that although the flash client endpoint has been reached, it hasn't answered the call.

### Result

**On success:** The API returns "true" if the legID is valid and the request has been forwarded to FMG.

**On failure:** The API returns "false".

## CallLegService.setLegStateSendReceive(legID)

You can use this API to request FMG to change the state of a call leg to SendRecv.

### Input

The input parameter, `legID`, is required only for incoming call legs for a flash phone (with `isOriginating = false`). The setup for audio communication is considered complete when the flash client accepts the call, subscribes to the stream, and starts publishing an audio stream. Following this set up, the API must request the SendRecv state.

For call legs with `isOriginating = true`, `leg.state.ring` and `leg.state.sendrecv` are automatically set by FMG, under the assumption that the flash phone has already published a stream to FMG.

### Result

**On success:** The API returns "true" if the `legID` is valid and the request has been forwarded to FMG.

***Note:** FMG starts processing the media only when this state has been reached. It is, therefore, essential for the incoming call legs to set this state when the flash client is ready to talk and has subscribed the stream.*

**On failure:** The API returns "false".

## CallLegService.setLegStateHangup(legID, cause)

You can use this API to request FMG to change the state of a call leg to Hangup, to end a call leg.

### Input

The input parameters for this API are as follows:

- `legID`: A currently active `legID`.
- `Cause`: An appropriate `causeID`.

### Result

**On success:** The API returns "true" if the `legID` is valid and the request has been forwarded to FMG.

**On failure:** The API returns "false".

## CallLegService.pushNewStream(legID, streamName)

You can use this API to indicate to FMG the `streamName` published by a flash phone for a call leg. This API may be invoked multiple times with unique audio stream names. When all such input streams are audio only, FMG would automatically perform audio mixing in realtime, and deliver the mixed audio output to the remote endpoint.

**Note:** For a `callLeg`, no action is taken by FMG when this API is invoked multiple times with same `<streamName>`. FMG would subscribe the `<streamName>` upon first unique invocation and subsequent API calls for the same `<streamName>` would be discarded.

### Input

The input parameters for this API are as follows:

- `legID`: A currently active legID.
- `streamName`: The stream that the flash client is publishing on this leg.

**Result**

**On success:** The API returns "true" if the legID is valid and the request has been forwarded to FMG.

**On failure:** The API returns "false".

*Note: FMG doesn't send any confirmation after successfully subscribing the input stream.*

## CallLegService.removeStream(legID, streamName)

You can use this API to indicate to FMG that the given stream is no longer published by flash client.

**Input**

The input parameters for this API are as follows:

- `legID`: A currently active legID.
- `streamName`: The stream that the flash client is publishing on this leg.

**Result**

**On success:** The API returns "true" if the legID is valid and the request has been forwarded to FMG.

**On failure:** The API returns "false".

*Note: FMG doesn't send any confirmation after successfully unsubscribing the input stream.*

## CallLegService.sendLegMessage(legID, info)

You can use this API to send DTMF digits over an active call leg.

**Input**

The input parameters for this API are as follows:

- `legID`: A currently active legID.
- `info`: To send DTMF digits, the info object should contain following properties:
  - `info.type`: "leg.message.type.dtmf"
  - `info.data`: string containing one or more DTMF characters

*Note: Valid DTMF characters are {[0-9], A, B, C, D, \*, #}.*

**Result**

**On success:** The API returns "true" if the message format is correct and the request has been forwarded to FMG.

**On failure:** The API returns "false".

*Note: FMG doesn't send any acknowledgment for the messages received.*

## CallLegService.setWriteCodec(legID, codecName)

You can use this API to set the codec of FMG to FMS stream for a call leg.

### Input

The input parameters for this API are as follows:

- `legID`: The legID whose codec is to be changed.
- `codecName`: String representing the codec type.

*Note: Supported string values are: "nm8k", "nm22k", and "speex".*

### Result

**On success:** The API returns "true" if the request has been forwarded to FMG.

**On failure:** The API returns "false".

## CallLegService.setSilenceLevel(legID, iVal)

You can use this API to set the `silenceLevel` property of an RTMP call leg. Silence is forced when the audio activity stays less than `SilenceLevel` for `silenceTimeout` milliseconds.

### Input

The input parameters for this API are as follows:

- `legID`: The legID whose `silenceLevel` property is to be changed.
- `iVal`: Integer representing the new `silenceLevel` property in percentage.

### Result

**On success:** The API returns "true" if the request has been forwarded to FMG.

**On failure:** The API returns "false".

## CallLegService.setWriteAudioGain(legID, iVal)

This API can be used to increase and decrease the audio gain of stream published by FMG.

### Input

The input parameters for this API are as follows:

`legID`: The legID whose `writeAudiodioGain` property is to be changed.

`iVal`: The desired integer gain value in the range [0, 100]. A gain value 50(default) means no change in the incoming audio. An input value 0 will completely suppress the audio. For other values in the range [1, 100] the decibel gain would vary from -10db to +10 db

### Result

**On success:** The API returns "true" if the request has been forwarded to FMG.

**On failure:** The API returns "false".

## CallLegService.setServiceID(strServiceID)

This API can be used replace the default value of serviceID with a custome value. The serviceID is also placed in `info.service` while invking any event handler.

### Input

The input parameters for this API are as follows:

- `strServiceID`: New serviceID value.

### Result

**On success:** The API returns "true" if the request has been forwarded to FMG.

**On failure:** The API returns "false".

## CallLegService.getServiceID()

This API can be used retrieve the current serviceID vakue associated with a CalllegServiceObject

### Result

**On success:** The API returns the serviceID value being used by the CallLegService Object..

**On failure:** The API returns "false".

# Chapter 4: Event handlers

Event Handlers are a set of interface functions that must be implemented by an FMS server-side application using leg service. FMG remotely invokes these APIs to pass event-related information to FMS server-side applications. Typically, these events are related to the state of a call leg or FMG performance.

## CallLegService.onStatus(info)

This event handler notifies the current state of FMG. It is invoked in any of the following cases:

- when FMG state changes
- when there is no activity between FMS and FMG

Parameter name	Description
info.type	Contains string value "legService.status.info".
info.data	<p>Contains any of the following values:</p> <ul style="list-style-type: none"> <li>• legService.status.init</li> <li>• legService.status.ready</li> <li>• legService.status.active</li> <li>• legService.status.busy</li> <li>• legService.status.blocked</li> </ul> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• FMG should be used for the creation of a new call leg when the status of leg service is either "ready" or "active".</li> <li>• The status of leg service changes according to the settings placed in the &lt;MaxCutoffCPUUsage&gt; tag in fmsg.xml.</li> <li>• FMG can also send self-triggered legService.status.blocked under extreme resource usage; for example, very high (&gt; 90%) CPU usage and memory consumption.</li> </ul>
info.metadata.profileID	(string) FMG profile ID of the leg service. This unique profile ID is the same as specified in rtmp.xml (if any); or else, it contains the FMG-generated profile ID whenever the <EnableAutoProfile> tag is enabled and the profile in use is not explicitly mentioned in rtmp.xml.
info.service	service ID of the SSAS leg service which invoked this handler instance.

## CallLegService.onLeg(info)

This event handler is to be implemented by the SSAS application using CallLegService. It is invoked when FMG has created a call leg object to a flash endpoint on the request of either a workflow (in the case of an incoming call) or the application itself (in the case of an outgoing call).

This custom handler returns "true" if it accepts the call and "false" otherwise. The leg will be rejected if no value is returned. Info object contains property-value information about this call leg.

Parameter name	Description
info.legID	Denotes the legID of this leg. The application stores this legID for future reference till this leg is active.
info.isOriginating	Denotes if the call leg is incoming or outgoing. If the value is "true", it indicates an outgoing call; if the value is "false", it indicates an incoming call.
info.refID	Contains the reference ID for an outgoing call request; value is null for incoming calls. This is the same ID that is received when createNewcall is invoked.
info.callerID	Contains the fully qualified address of the originator of the call. For outgoing calls, this value is used to map with the FMS client (flash phone).
info.calleeID	Contains the fully qualified address of the recipient of the call. For incoming calls, this value is used to map with the Flash client (flash phone).
info.incomingStream	Contains the streamName that should be subscribed by an RTMP phone for incoming media.
info.callerIDName	Denotes the display name for the caller ID panel.
info.callerIDNumber	Denotes the display number for the caller ID panel.
info.service	service ID of the SSAS leg service which invoked this handler instance.
info.isPrivacyEnabled	Specifies whether caller has requested privacy(true) or not (false).

## CallLegService.onLegStatus(info)

This event handler is to be implemented by SSAS application using CallLegService. It is invoked when FMG changes the state of a call leg associated with this application instance, either by itself or due to a request made by the SSAS application. No return value is expected.

Parameter name	Description
info.legID	Indicates the legID of the call leg.
info.status	Indicates the new state of the call leg. It contains any of the following values: <ul style="list-style-type: none"><li>leg.state.init</li><li>leg.state.ring</li><li>leg.state.sendrecv</li><li>leg.state.execute</li><li>leg.state.hold</li><li>leg.state.reset</li><li>leg.state.hangup</li></ul>

Parameter name	Description
info.service	service ID of the SSAS leg service which invoked this handler instance.
info.cause	Contains the number containing the code of hangup cause (cause code), when the new state is hangup.
info.causeString	(string) Contains the description of the cause code returned by the info.cause parameter.

## CallLegService.onLegMessage(info)

This event handler is to be implemented by SSAS application using CallLegService. It is invoked when FMG sends an informational message on a call leg. No return value is expected.

Parameter name	Description
info.legID	Indicates the legID of the destination call leg for this message.
info.service	service ID of the SSAS leg service which invoked this handler instance.
info.type	Indicates the type/class of the message. The possible values depend on the type of message: <ul style="list-style-type: none"><li>"leg.message.type.indication": for leg-specific indications in info.data.</li><li>"leg.message.type.dtmf": for DTMF digits sent by remote leg/FMG to the endpoint associated with this call leg.</li></ul>
info.data	Stores the actual content of messages. Some examples are: <ul style="list-style-type: none"><li>"message.indication.progress" indication is received when FMG starts processing a call leg as part of the workflow.</li><li>"message.indication.remoteRinging" indication is received when the remote-side is in ring state during a bridged call.</li><li>"message.indication.remoteAnswer" indication is received when, during a bridged call, the remote-end has answered. Additional properties of "message.indication.remoteAnswer" include "info.metadata.remoteLegID", which, during a bridged call, contains the legID of the remote leg associated with the current leg.</li><li>For leg.message.type.dtmf, info.data contains the DTMF digit(s) received by the call leg.</li></ul>

## CallLegService.onError(info)

This event handler is to be implemented by SSAS application using CallLegService. It is invoked when FMG sends an error message to the CalllegService object. No return value is expected.

Parameter name	Description
info.data	(if any) Indicates the referenceID for the request to which this error corresponds.
info.service	service ID of the SSAS leg service which invoked this handler instance.
info.type	Indicates the type/class of the message. The possible values depend on the type of message: <ul style="list-style-type: none"><li>"legService.error.createCall": Indicates that the error corresponds to the failure in the execution of CallLegService.createCall API.</li></ul>



Parameter name	Description
info.cause	the error cause ID
info.causeString	The string representation of the error cause
info.metadata	Stores the other relevant properties/data to indicate the parameters leading to the error condition.

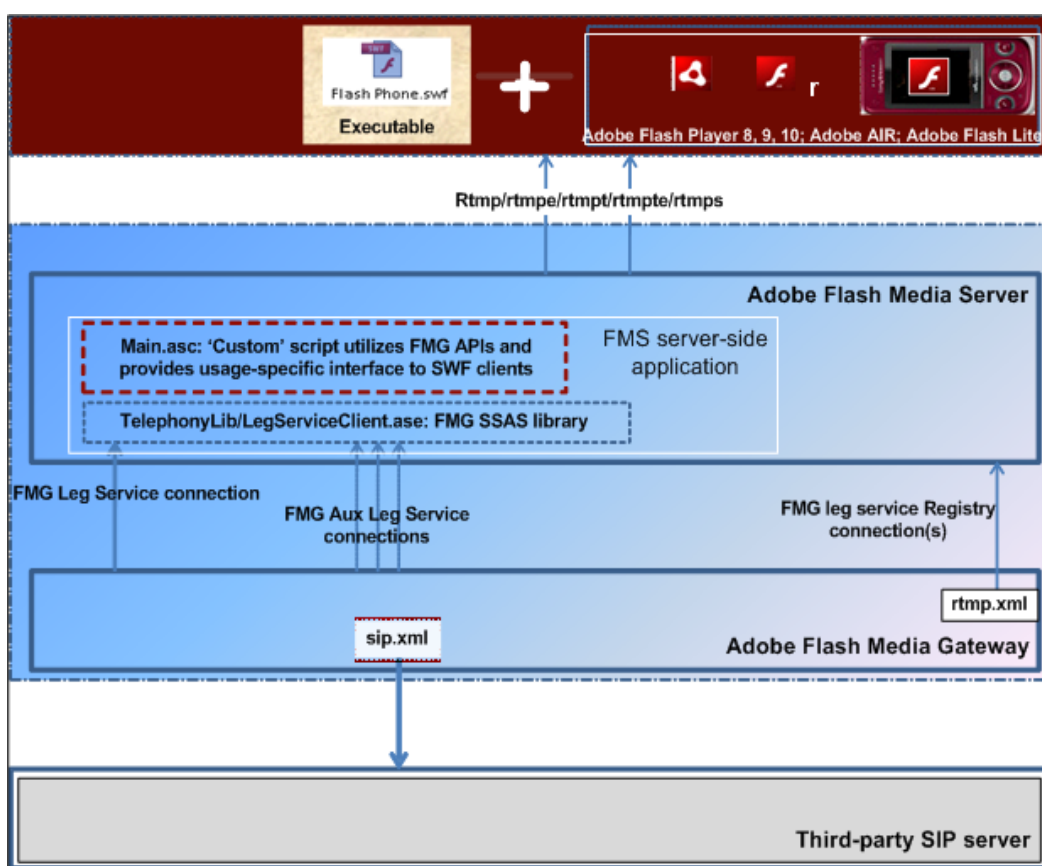
# Chapter 5: Sample usage

## Overview

Enabling flash player VoIP using FMG requires:

- An FMS server-side application, built using the FMG leg service APIs
- A client-side SWF executable

The following illustration shows the various participants of a simple FMG-based VoIP infrastructure, with the customized FMS server application and client-side SWF highlighted.



## Creating a telephony application using FMG leg service API

Server-side leg service APIs are used to access the FMG. An application might require extending the access of FMG APIs to a flash client, using FMS Remote Method invocations such as `NetConnection.call` at the client-side and `Client.call` at the server-side.

## Leg service initialization

- 1 Create a new FMS application with server-side script file (main.asc).
- 2 Copy the TelephonyLib folder to this new application.
- 3 Load FMG SSAS Library in main.asc by using:

```
load("TelephonyLib/callLegServiceClient.asc");
var myLegService = new CallLegService();
```

- 4 When FMG has started with appropriate leg service registry settings in rtmp.xml, request FMG leg service by using `services.request(<leg-service-name>)`.

For requesting a leg service named “telephony”, the following function call can be used.

```
services.request("telephony");
```

- 5 Wait for FMG leg service to connect and activate the `CallLegService` object by passing FMG leg service connection’s client object.

```
myLegService.setServer(client);
```

- 6 If `<AuxConnectionPool>` is enabled in `rtmp.xml` (recommended), add `application.onConnect` and `application.onDisconnect` to allow connects and disconnects from the clients with `client.agent` property set to FMG Aux Leg Service 1.0.

For details and working source code, see the previous sections of this document and the sample telephony application included in FMG installation. The sample telephony application can be found at `<FMG-Install-Dir>/Flash Media Gateway/FMSApplications/telephony/`.

All leg service connections from FMG are made with the extra parameters- `serviceName` and `passcode`. `Passcode` should be as specified in the FMG configuration file, `rtmp.xml`. Developers can implement their own authentication routine in `application.onConnect` to accept or reject the connection. This helps in meeting both the identification and authentication requirements of FMG service connections. It is also quicker, as decision can be made when a connection is initiated.

```
application.onConnect = function(client, serviceName, passCode)
{
    // A new client is connected. Authenticate and identify this client and take suitable
    // actions from subscribing to an unauthorized stream.
    if(client.agent == "FMG Aux Leg Service 1.0")
    {
        // If we are sure that it is FMG Aux Leg Service then let it connect.
        trace("FMG Aux Leg Service 1.0");
        trace('serviceName: ' + serviceName);
        trace('passCode: ' + passCode);
        return;
    }
}
```

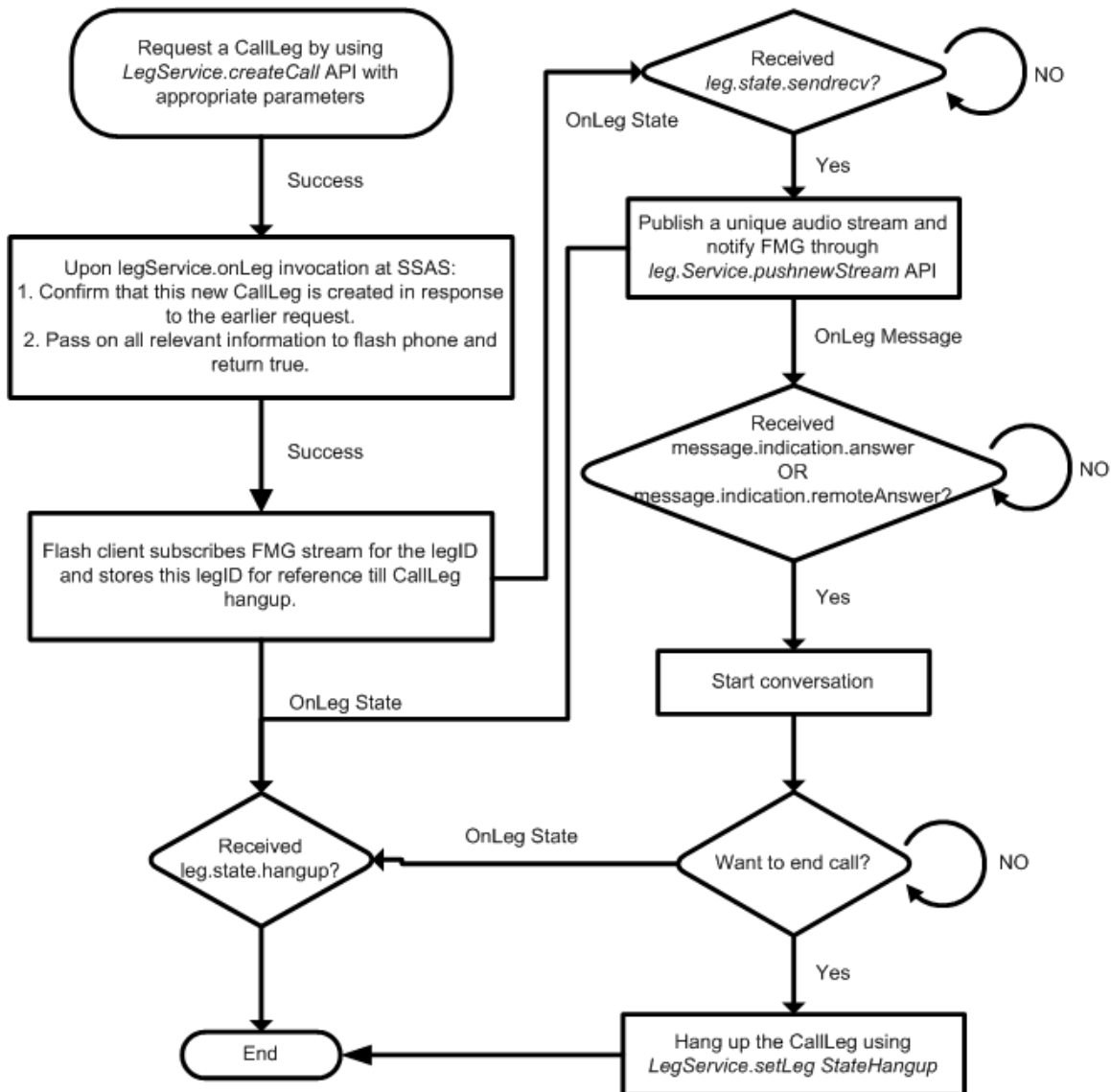
## Incoming/outgoing calls using FMG leg service API

To create and manage call legs, the SSAS application requires implementing a phone number-based addressing mechanism for its flash clients, leg service handler routines, and so on. For further details, see the sample telephony application that includes FMS server-side application source and client-side scripts.

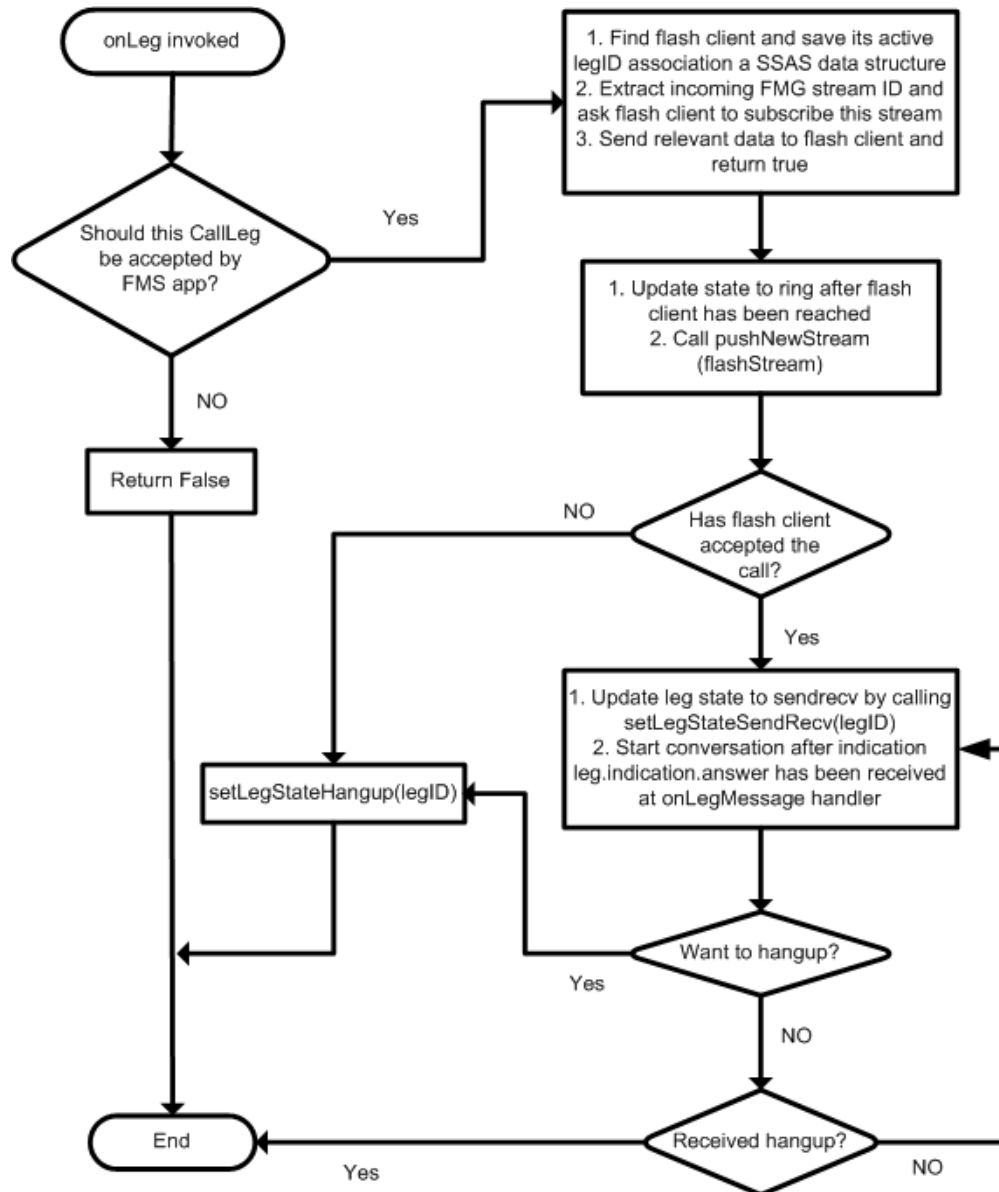
**Note:** The pre-requisite for an SSAS application to address its phones is that the address string must contain only alpha-numerical data.

The following diagrams provide a snapshot of how a sample telephony application uses a sequence of API calls to manage a call from creation till hangup.

**Decision flow diagram for an outgoing call using FMG leg service API**



Decision flow diagram for an incoming call using FMG leg service API



## Managing call leg states

For incoming call legs, with property isOriginating = false:

- When “true” is returned from onLeg handler, the state is automatically set to leg.state.init.
- When a flash client is available and hasn’t answered yet, the leg state must be set to leg.state.ring using setLegStateRing().

*Note: This is an optional step.*

- When a call is accepted from flash client, the state must be set to leg.state.sendrecv. message.indication.answer indicates that call is now fully connected and ready for conversation.

- When FMG receives `leg.state.hangup`, it indicates termination of the call.

For outgoing calls, with property `isOriginating = true`:

- When “true” is returned from `onLeg` handler, the state is set to `leg.state.init`. FMG takes over state control.
- When FMG receives `leg.state.sendrecv` and `message.indication.answer`, it indicates a successfully established call.
- When FMG sends `message.indication.remoteAnswer` and `message.indication.remoteRinging`, it indicates the status of a remote party (if any).
- When FMG receives `leg.state.hangup`, it indicates termination of the call.

## Best practices

- In production deployment, it is recommended to enable the `rtmp.xml` tag, `<AuxConnectionPool>`, with appropriate settings. This feature is especially useful when more than 5-10 calls are created using one leg service connection. By default, this tag is set to false (disabled).
- To reduce echo and maintain smooth capture of voice generated from a Flash Player-based client, customized automatic microphone gain leveling routine should be implemented at client scripts. For sample source code, see the sample Flash phone included in the FMG installation.  
At the Flash player, while encoding in Speex, microphone gain should regulated such that the maximum audio activity level stays below 60. For NM, encoded audio quality stays good at around activity level below 70.
- While subscribing Live Audio stream for real-time communication, it is recommended not to set buffer using `NetStream.setBufferTime()`.
- To reduce delay caused by publisher-side buffering at FMS, it is recommended to disable live queues in the FMS configurations file, `application.xml`.

```
<Application>
  <StreamManager>
    <Live>
      <!-- A message queue is used to buffer the incoming messages from the publisher -->
      <!-- so that the server can send messages in chunks to the subscribers. Queuing -->
      <!-- can be disabled so that individual messages will be sent out immediately to -->
      <!-- the subscribers. The "enabled" attribute can be set to "false" to disable -->
      <!-- queuing. -->
      <Queue enabled="false">
    </Queue>
  </Live>
</StreamManager>
</Application>
```