

ADOBE® FLASH® MEDIA GATEWAY 2.0

HTTP API REFERENCE

© 2010 Adobe Systems Incorporated. All rights reserved.

Adobe® Flash® Media Gateway HTTP 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, and Flash 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

Acronyms	1
Chapter 1: Introduction	2
Chapter 2: HTTP API Summary	3
Chapter 3: HTTP Get/Set APIs	4
getFMGStatus	4
getPluginStatus	5
changePluginConf.....	7
stopFMG	8
getCallDetails.....	8
Chapter 4: HTTP Configuration APIs	10
FMGGetOID	10
FMGSetOIDValue	12
FMGSetOIDAttrib	13
FMGSaveConfig	13
FMGRevertConfigChanges	14

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

Adobe® Flash® Media Gateway (FMG) provides a HTTP interface to perform tasks such as:

- Displaying the status of FMG. This includes details such as CPU usage, up-time, current call leg count, and list/details of the registration services provided to different FMS.
- Restarting FMG on demand.
- Adding/deleting leg/control service dynamically and maintaining the changes when FMG restarts.
- Authenticating at multiple levels, to authorize the usage of HTTP interface (set in http.xml).
- Viewing/editing XML configuration in the FMG interface over HTTP.

The FMG HTTP interface listens on port 2222 by default.

For more information on the configuration options related to the HTTP interface (http.xml), see *Adobe® Flash® Media Gateway Installation and Configuration Guide*.

Chapter 2: HTTP API Summary

API method	Description
getFMGStatus	Gets status of the FMG, including statistics such as uptime, CPU usage, and memory usage.
getPluginStatus	Retrieves RTMP/SIP plugin status such as connection and remote entities status.
changePluginConf	Provides a generic interface for the services provided by plugins. For more details on the APIs, see the description of specific APIs provided by the plugins.
stopFMG	Stops/restarts FMG with the ability to wait for the existing calls to finish, or force terminate them.
getCallDetails	Lists the hangup cause and the total number of call legs that were hungup due to that cause.
FMGGetOID	Retrieves any XML tag, along with its child tags, from the configuration XML files.
FMGSetOIDValue	Provides an interface to add/edit any XML tag value in the configuration XML files.
FMGSetOIDAttrib	Provides an interface to add/edit any XML tag attribute in the configuration XML files.
FMGSaveConfig	Provides an interface to save any changes made to the configuration XML files.
FMGRevertConfigChanges	Reverts all changes made using the HTTP Configuration APIs since the last save.

Chapter 3: HTTP Get/Set APIs

Note: The values of *auser* and *apswd* that must be passed in the commands listed below, must match the username and password entered at the time of installation. These details are saved in the *fmg.ini* file.

getFMGStatus

You can use this API to retrieve the status and usage statistics for the FMG process.

Command: `http://fmghost:2222/admin/getFMGStatus?auser=username&apswd=password`

Result if successful

```
<?xml version="1.0" encoding="utf-8"?>
<result>
  <level> status </level>
  <code> fmg.request.success </code>
  <timestamp> 7/23/2007 9:10:37 AM GMT </timestamp>
  <data>
    <cpuUsage> 3 </cpuUsage>
    <totalCPUUsage> 5 </totalCPUUsage>
    <memoryLoad> 3 </memoryLoad>
    <physicalMemUsage> 3 </physicalMemUsage>
    <uptime> 2:40:04 <uptime>
    <totalCallLegs> 20 </totalCallLegs>
    <threads> 3 </threads>
    <handles> 100 </handles>
    <status> NORMAL/BUSY/BLOCKED </status>
  </data>
</result>
```

where:

- `cpuUsage`: CPU usage for FMG process (average of last 5 secs), in percentage.
- `totalCPUUsage`: CPU usage for FMG server machine (average of last 5 secs), in percentage.
- `memoryLoad`: Memory load of the FMG process, in percentage.
- `physicalMemUsage`: Working set size of the FMG process, in MB.
- `uptime`: Uptime of FMG. When the uptime is less than 24 hrs, the format is HH hours MM mins SS secs, where:
 - HH = no. of hours passed
 - MM = no. of minutes passed
 - SS = no. of seconds passed

When the uptime is greater than 24 hours, the format is D day(s) HH hours MM mins SS secs, where:

- D = no. of days passed
- HH = no. of hours passed
- MM = no. of minutes passed

- SS = no. of seconds passed
- status: Can be one of the following:
 - NORMAL when FMG is ready to service leg requests.
 - BUSY when min cpu/memory thresholds are reached and FMG is using high resources.
 - BLOCKED when max cpu/memory thresholds are reached and FMG can't service any more requests.
- totalCallLegs: Number of existing call legs running in FMG.
- threads: Total number of threads for the FMG process.
- handles: Total number of handles for the FMG process.

Result if unsuccessful

```
<?xml version="1.0" encoding="utf-8"?>
<result>
  <level> error </level>
  <code> fmg.request.rejected </code>
  <description> Admin user requires valid username and password. </description>
  <timestamp> 11/26/2008 2:40:04 PM GMT </timestamp>
</result>
```

getPluginStatus

You can use this API to retrieve plugin-specific status. For RTMP plugins, it lists the following:

- Leg and control service statuses
- The count of leg services connected
- URLs to which the leg services are connected

Command:

http://fmghost:2222/admin/getPluginStatus?auser=username&apswd=password&plugin=<rtmp or sip>

Result for RTMP plugin, if unsuccessful

```
<?xml version="1.0" encoding="utf-8"?>
<result>
<level> status </level>
<code> fmg.request.success </code>
<timestamp> 7/23/2007 9:10:37 AM GMT </timestamp>
<data>
  <rtmp>
    </status>
    <Registrations>
    <LegService>
      <count> </count>
      <_0>
        <Host> </Host>
        <ServiceName> </ServiceName>
        <Status> </Status>
      </_0>
      <_1>
```



```

...
...
...
</_1>
</LegService>
  <ControlService>
    <count> </count>
    <_0>
      <Host> </Host>
      <ServiceName> </ServiceName>
      <Status> </Status>
    </_0>
    <_1>
      ...
      ...
      ...
    </_1>
  </ControlService>
</Registrations>
<legCount> </legCount>
<legServicesCount> </legServicesCount>
<LegServices>
<_0>
<Url>rtmp://localhost/telephony/john</Url>
</_0>
</LegServices>
  </status>
</rtmp>
</data>
</result>

```

Result for SIP plugin, if unsuccessful

```

<?xml version="1.0" encoding="utf-8"?>
<result>
<level> status </level>
<code> fmg.request.success </code>
<timestamp> 7/23/2007 9:10:37 AM GMT </timestamp>
<data>
  <sip>
    <Status>
      <Registrations>
        <Profiles>
          <Profile>
            <ProfileID> asterisk </ProfileID>
            <UserName> john </UserName>
            <Host> 10.40.60.207:5060 </Host>
            <Context> sipToSipCall </Context>
            <RegistrationStatus> Registered </RegistrationStatus>
            <Availability> Up </Availability>
          </Profile>
          <Profile>
            <ProfileID> sipAPhone </ProfileID>
            <UserName> 777 </UserName>
            <Host> 127.0.0.1:5060 </Host>
            <Context> sipPhoneContext </Context>

```

```

        <RegistrationStatus> Registered </RegistrationStatus>
        <Availability> Down </Availability>
    </Profile>
    <Profile>
        <ProfileID> sipGPhone </ProfileID>
        <UserName> 777 </UserName>
        <Host> localhost:5060 </Host>
        <Context> sipGatewayContext </Context>
        <RegistrationStatus> Registered </RegistrationStatus>
        <Availability> Down </Availability>
    </Profile>
</Profiles>
</Registrations>
</Status>
</sip>
</data>
</result>

```

changePluginConf

You can use this API to create a new RTMP leg service or a control service to an application on FMS.

Command:

http://fmghost:2222/admin/changePluginConf?auser=username&apswd=password&plugin=rtmp&args=createService,<serviceType>,rtmp://myserver/myapp/myinstance,passcode_string,service_name
where:

- args denotes the createService command.
- <serviceType> is the service connection type. It can have the values- legService or controlService.
- <passcode_string>, <service_name> are the respective values which FMG will use to get authenticated or /and to get uniquely identified on FMS SSAS. Uniqueness of service_name is to be ensured by admin/developer as per application requirement. These parameters are applicable only for <serviceType> = 'legService'
- rtmp://myserver/myapp/myinstance is the RTMP URL of the FMS application to which service connection must be made.

Result if successful

```

<?xml version="1.0" encoding="utf-8"?>
<result>
    <level> status </level>
    <code> fmg.request.success </code>
    <timestamp> 7/23/2007 9:10:37 AM GMT </timestamp>
    <!-- data node will be empty -->
    <data> </data>
</result>

```

Result if unsuccessful

```

<result>
    <level> error </level>
    <code> FMG.Admin.CommandFailed </code>
    <timestamp> 5/14/2009 6:27:36 PM </timestamp>
    <description> Error Description </description>
</result>

```

stopFMG

You can use this API to stop or restart FMG.

Command:

`http://fmghost:2222/admin/stopFMG?auser=username&apswd=password&restart=true&wait=true`

When `wait=true` and `restart=true`, FMG stops creating new call legs and waits for all existing calls to get cleared before restarting FMG.

When `wait=false` and `restart=true`, FMG restarts immediately. When `restart=false`, FMG attempts to shut down either immediately or after all calls are cleared, based on the value of the `wait` parameter. After FMG has shut down, it, stops responding at the HTTP interface.

Result if successful

```
<?xml version="1.0" encoding="utf-8"?>
<result>
  <level> status </level>
  <code> fmg.request.success </code>
  <timestamp> 7/23/2007 9:10:37 AM GMT </timestamp>
  <!-- data node will be empty -->
  <data> </data>
</result>
```

Result if unsuccessful

```
<result>
  <level> error </level>
  <code> FMG.Admin.CommandFailed </code>
  <timestamp> 5/14/2009 6:27:36 PM </timestamp>
  <description> Error Description </description>
</result>
```

getCallDetails

You can use this API to get the call details (success/failure) since the start of FMG.

Command: `http://fmghost:2222/admin/getCallDetails?auser=username&apswd=password`

Result if successful

```
<?xml version="1.0" encoding="utf-8"?>
<result>
  <level> status </level>
  <code> fmg.request.success </code>
  <timestamp> 7/23/2007 9:10:37 AM GMT </timestamp>
  <data>
  </data>
</result>
```

Result if unsuccessful

```
<result>
  <level> error </level>
  <code> FMG.Admin.CommandFailed </code>
```

```
<timestamp> 5/14/2009 6:27:36 PM </timestamp>  
<description> Error Description </description>  
</result>
```

Chapter 4: HTTP Configuration APIs

The Get/Set interface doesn't make the changes effective in the FMG or permanent in the XML files. Use the FMGSaveConfig interface to save to files. Then, reload/restart the FMG to make the changes effective.

Note: Some XML tags require a restart of FMG for the changes to take effect; some require only a reload.

- Reload uses FMGCommand, with `reload` as the parameter.
- Restart uses FMGStop, with `restart=true`.

For more information on Get/Set interface, see [HTTP Get/Set APIs](#).

The FMG Configuration Get/Set interface is for the following XML files:

- `fmsg.xml`
- `workflow.xml`
- `sip.xml`
- `rtmp.xml`

All XML tags in the FMG XML configuration files are identified by unique Object-IDs (OID). For example, FMG.SIP identifies the entire SIP xml file. However, FMG.SIP.FMSGConfig.portUpperLimit identifies only the tag, `<portUpperLimit>`, in `sip.xml`. Similarly, FMG.RTMP.AuxConnectionPool.LoadPerConnection identifies a unique tag in the `rtmp.xml` file.

There are instances where a single XML tag contains multiple XML entries with the same name. For example, FMG.SIP.Profiles contains more than one `<Profile>` entries. To identify a unique profile in such cases, the OID must be assigned a suffix that locates the individual `<Profile>` per its position in the `<Profiles>` tag. For example, in the OID, FMG.SIP.Profiles.Profile.3, the suffix "3" helps to locate the third `<Profile>` entry in the `<Profiles>` tag. If the number suffix is omitted, it is assumed that the OID points to the first entry. If the number is immediately greater than the count of the same tag entries, it is assumed that a new tag entry is being requested.

Note: OIDs are case-dependent.

FMGGetOID

You can use this API to retrieve any XML tag and all its child tags.

Example 1:

Command:

```
http://fmghost:2222/admin/FMGGetOID?auser=admin&apswd=abc123&oid=FMG.sip.Profiles.Profile.2
```

Result if successful

```
<result>
<level> status </level>
<code> FMG.Call.Success </code>
<timestamp> 5/13/2009 4:29:30 PM </timestamp>
-
```

```

    <data>
-
    <result>
-
        <Profile>
        <profileID> Gibraltar2 </profileID>
        <userName> 300 </userName>
        <password> password </password>
        <displayName> 300 </displayName>
        <registrarAddress> 10.40.63.164:9000 </registrarAddress>
        <doRegister> 1 </doRegister>
        <defaultHost> 10.40.63.164 </defaultHost>
        <hostPort> 0 </hostPort>
        <context> sipToSipCall </context>
        <supportedCodecs/>
        </Profile>
    </result>
    </data>
</result>

```

Result if unsuccessful

```

<result>
    <level> error </level>
    <code> FMG.Admin.CommandFailed </code>
    <timestamp> 5/14/2009 6:27:36 PM </timestamp>
    <description> Error Description </description>
</result>

```

Example 2:**Command:**

http://fmghost:2222/admin/FMGGetOID?auser=admin&apswd=abc123&oid=FMG.SIP.FMSGConfig.SipPort

Result if successful

```

<result>
    <level> status </level>
    <code> FMG.Call.Success </code>
    <timestamp> 5/13/2009 5:26:23 PM </timestamp>
-
    <data>
-
    <result>
        <SipPort> 5678 </SipPort>
    </result>
    </data>
</result>

```

Result if unsuccessful

```
<result>
  <level> error </level>
  <code> FMG.Admin.CommandFailed </code>
  <timestamp> 5/14/2009 6:27:36 PM </timestamp>
  <description> Error Description </description>
</result>
```

FMGSetOIDValue

You can use this API to retrieve any XML tag value.

XML tags can only be edited/added one at a time. For example, to edit multiple entries in the third listed profile in sip.xml (FMG.SIP.Profiles.Profile.3), multiple FMGSetOIDValue calls (such as FMG.SIP.Profiles.Profile.3.ProfileID and FMG.SIP.Profiles.Profile.3.userName) are required for each tag entry. The interface maintains the case-sensitivity of the tags.

FMGSetOIDValue can also be used for adding new XML tags.

To add a new tag, ensure that the last string of the OID corresponds to the XML tag to be added. For example, to add a new tag named <NewTag> in the FSMGConfig section of the sip.xml file, the OID should be FMG.SIP.FMGMSConfig.NewTag and the value must represent the value of <NewTag>.

Sometimes, a tag has multiple entries. For example, FMG.SIP.Profiles contains more than one <Profile> entry. To add a new <Profile> entry, the OID must be an increment of the total count of the Profiles present. For example, if there are three Profiles in the sip.xml, the OID FMG.SIP.Profiles.Profile.4 will add a new <Profile> tag into the xml file. As Profiles contains multiple entries, add one entry at a time.

When successful, the <tagStatus> in the returned XML denotes whether a restart or a reload is required for the changes to take effect. The values in <tagStatus> can be ReloadRequired or RestartRequired.

FMGSaveConfig must be issued to save the changes.

Command:

http://fmghost:2222/admin/FMGSetOIDValue?auser=admin&apswd=abc123&oid=FMG.sip.Profiles.Profile.2.supportedCodecs.codecId&value=Speex
where:

- oid = OID for the XML tag
- value = Value to be set for the XML tag (optional)

Result if successful

```
<result>
  <level> status </level>
  <code> FMG.Call.Success </code>
  <timestamp> 5/14/2009 5:38:31 PM </timestamp>
  <data>
    <tagStatus> ReloadRequired </tagStatus>
  </data>
</result>
```

Result if unsuccessful

```
<result>
  <level> error </level>
  <code> FMG.Admin.CommandFailed </code>
  <timestamp> 5/14/2009 6:27:36 PM </timestamp>
  <description> Error Description </description>
</result>
```

FMGSetOIDAttrib

You can use this API to edit attribute values in the XML tags. The OID should adhere to the naming convention so that each OID identifies a unique profile.

Note: To set and edit attributes for a new XML tag, it should first be added using *FMGSetOIDValue*.

The values in the <tagStatus> tag can be ReloadRequired or RestartRequired.

Note: *FMGSaveConfig* must be issued to save the changes.

Command:

`http://fmghost:2222/admin/FMGSetOIDAttrib?auser=admin&apswd=abc123&oid=FMG.rtmp.Registrations.LegService.Server.2&attrib=host&value=127.0.0.1`

Result if successful

```
<result>
  <level> status </level>
  <code> FMG.Call.Success </code>
  <timestamp> 5/14/2009 5:38:31 PM </timestamp>
  <data>
    <tagStatus> RestartRequired </tagStatus>
  </data>
</result>
```

Result if unsuccessful

```
<result>
  <level> error </level>
  <code> FMG.Admin.CommandFailed </code>
  <timestamp> 5/14/2009 6:27:36 PM </timestamp>
  <description> Error Description </description>
</result>
```

FMGSaveConfig

You can use this API to save the XML configuration file to the disk. This does not load the configuration into FMG; the reload FMG command has to be issued for the changes to take effect. The OID must indicate either the XML file to be saved (FMG.sip or FMG.rtmp) or to save all (FMG).

Command: `http://fmghost:2222/admin/FMGSaveConfig?auser=admin&apswd=abc123&oid=FMG.rtmp`

Result if successful

```
result>
  <level> status </level>
  <code> FMG.Call.Success </code>
  <timestamp> 5/13/2009 5:15:53 PM </timestamp>
</result>
```

Result if unsuccessful

```
<result>
  <level> error </level>
  <code> FMG.Admin.CommandFailed </code>
  <timestamp> 5/14/2009 6:27:36 PM </timestamp>
  <description> Error Description </description>
</result>
```

FMGRevertConfigChanges

You can use this API to revert/undo any changes made to the XML using the Set commands. This API reverts all the changes made to the in-memory configuration since the last save.

Command:

<http://fmghost:2222/admin/FMGRevertConfigChanges?auser=admin&apswd=abc123&oid=FMG.rtmp>

Result if successful

```
<result>
  <level> status </level>
  <code> FMG.Call.Success </code>
  <timestamp> 5/13/2009 5:15:53 PM </timestamp>
</result>
```

Result if unsuccessful

```
<result>
  <level> error </level>
  <code> FMG.Admin.CommandFailed </code>
  <timestamp> 5/14/2009 6:27:36 PM </timestamp>
  <description> Error Description </description>
</result>
```