SMG SOAP Interface

Specification

XSPC-0002-00

February 2012

Notice

While XOR Media believes the information included in this publication is correct as of the publication date, information in this document is subject to change without notice.

UNLESS EXPRESSLY SET FORTH IN A WRITTEN AGREEMENT SIGNED BY AN AUTHORIZED REPRESENTATIVE OF XOR MEDIA LTD., XOR MEDIA MAKES NO WARRANTY OR REPRESENTATION OF ANY KIND WITH RESPECT TO THE INFORMATION CONTAINED HEREIN, INCLUDING WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PURPOSE. XOR Media Ltd. assumes no responsibility or obligation of any kind for any errors contained herein or in connection with the furnishing, performance, or use of this document.

Software described in XOR Media documents (a) is the property of XOR Media Ltd. or the third party, (b) is furnished only under license, and (c) may be copied or used only as expressly permitted under the terms of the license.

Manual Title: *SMG SOAP Interface Specification*
Revision: 2.12.002
Printing Date: February 2012

Published by **XOR Media Ltd**.

Table of Contents

Preface

Purpose

This guide describes the specification of XOR Media SMG SOAP Interface.

The specification is aimed for the integration with XOR Media MediaGateway Pro system by third-party applications and is extensible for new task type.

Audience

This guide is intended for use by software developers for controlling applications for SMG system.

Related Documents

MediaGateway Pro Installation Guide

MediaGateway Pro User's Guide

SOAP Version 1.2 Specification http://www.w3.org/TR/soap12/

Revision History

Below table describes the major revisions that have been made to this manual.

Revision	Date	Description
2.12.001	2011-7-13	Created
2.12.002	2012-2-13	Added SMGGatewayService port number 30005.

1 Overview

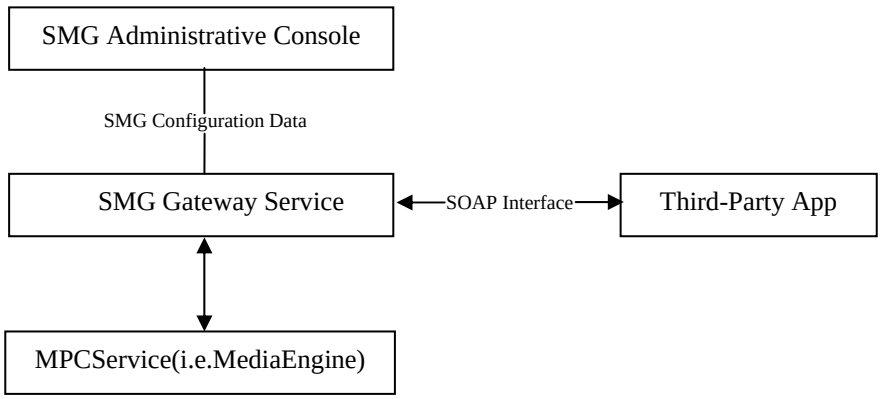
The MediaGateway (SMG) is a XOR Media developed software product that provides a highly effective solution to conversion among many different media formats. The SMG application brings a whole new level of simplification in workflow designs and managed transfers between the different generation of XOR Media servers. It also enables much simpler integrations between various third-party productions, MAM, and archive systems.

By using MediaGateway, a range of workflow and content management functions can be handled, including Ingests, Transfers and Archiving, Transcoding. The most recent Web Service API interface allows third-party application Automation or MAM systems to implement direct control over services that are available with MediaGateway. SMG has made possible the deployment of tapeless broadcast workflows.

Basically execution of job in system of third-party application's integration with SMG (requiring version 3.2.1.5 or above) through SOAP interface can be in the form of (manual) policy type or task type. This guide addresses solely policy type execution form.

Within this framework of architecture only two of the SMG components, that is Administrative Console and Media Engine, are essential and required for implementing the integration. There is no need to install component Media Monitor or SMG Transfer Utility at the time of deploying SMG for integration.

A service component called SMG Gateway Service in this solution plays the vital role in receiving commands from web client and translating them before triggering MediaEngine to execute job. Ultimately the commands are processed by another service component called MPCService (i.e. Media Engine as seen in Administrative Console) waiting as an agent for commands from Gateway Service and executing them, additionally returning status upon receiving query request. The interactions between them are illustrated in the following diagram.



The entire workflow of SMG integration through SOAP interface for ingest or conversion can be boiled down to the following four major pieces.

1. SMG Administrative Console

   Configure SMG by adding devices, Media Engine, and manual policy.

2. SMG Gateway Service

   3rd application makes function calls via SOAP interface to create jobs for a manual policy.

   a. Create SOAP Server framework.

   b. Receive message from 3rd application via SOAP interface. The service port of SMGGatewayService is 30005.

   c. Parse message and translate to task info for MediaEngine.

   d. Manage task queue for different client.

   e. Send reply message to web client.

3. MediaEngine

   a. Execute the jobs.

   b. Report status and progress of the jobs when receiving query request.

4. 3rd Application

   a. Query the execution status of the jobs after having made function calls earlier for creating jobs.

Further discussion is given in detail in the SMG Setup and SOAP Functions Call respectively.

1.1 SMG Setup

Before integrating with SMG through SOAP SMG should be set up as the following.

1. Install SMG application on SMG server. See *SMG Pro Installation Guide* for details. Note that only components Administrative Console and Media Engine are required for SOAP integration. You do not need component Media Monitor or SMG Transfer Utility for SOAP.
2. Have a valid SMG license/license key plugged in SMG server.
3. Open SMG Administrative Console by running SMG application (SeaChangeMediaGateway.exe).
4. Add devices. See *SMG Pro User's Guide* for details.
5. Add and activate Media Engine.
6. Add manual policy for file ingest or conversion as you need.

1.2 SOAP Functions Call

To create and execute through SOAP interface jobs defined by manual policy in SMG, please follow the guidelines listed below.

All input and output parameters of the functions take the form of XML files. For a sample of those XML files as well as syntax of those interface functions please reference 2-SOAP Interface Functions for details. Some of the fields in the sample XML files must be substituted by the specific value in your particular case. Those that must be substituted are spelled out in the Remarks at the end of each function.

1. Call function **GetConfiguration()** to get the policy type (in this case always manual type) and a list of policy names.
2. Call function **CreateTask()** to create job and trigger its execution by providing parameters with name of your source file for ingest or conversion as well as the policy name returned from **GetConfiguration()** call.

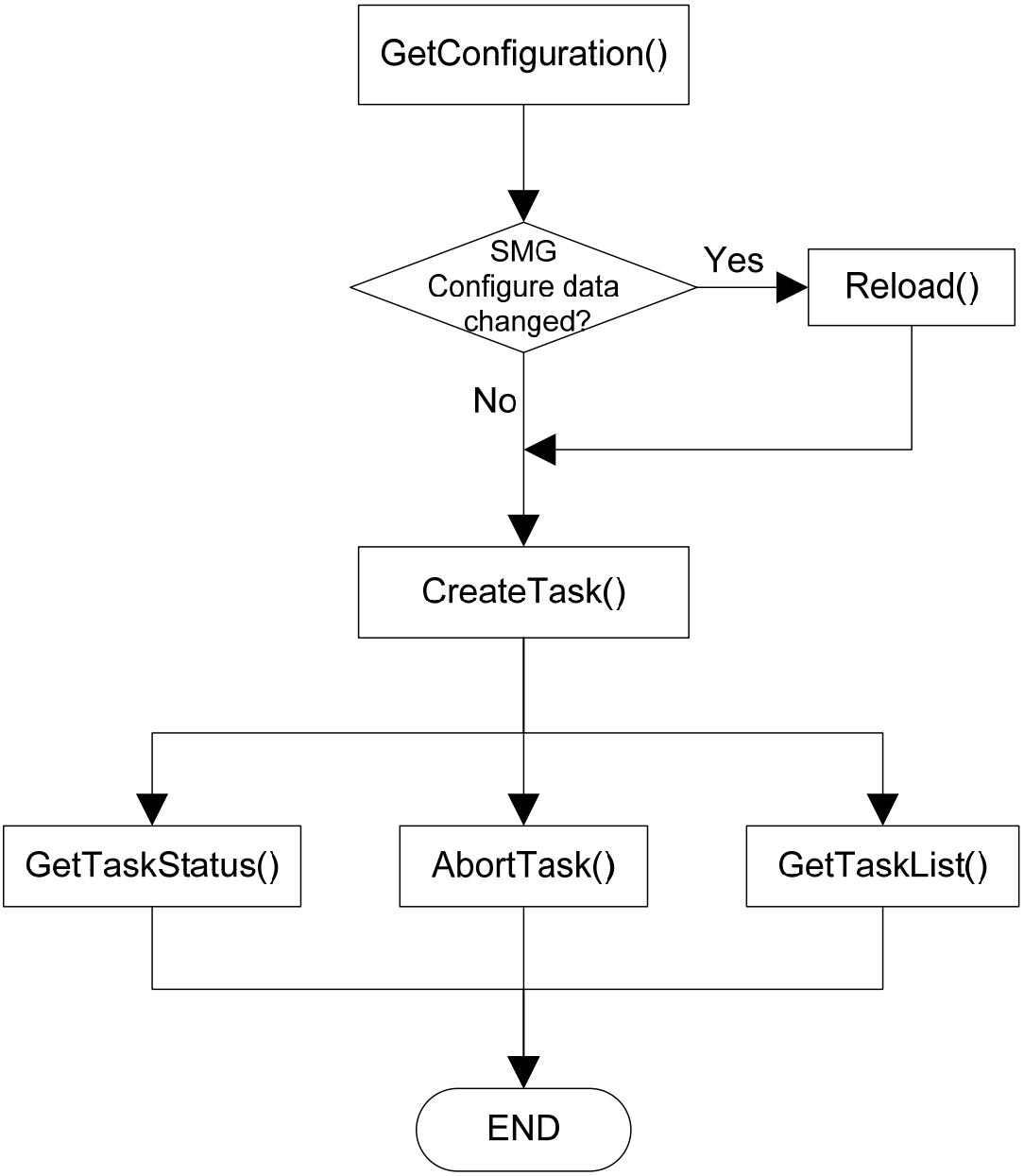
   If any time during that period SMG configuration has changed you have to call **Reload()** to get the most updated SMG configuration before creating and executing jobs by calling **CreateTask()**.

   Note jobs created by SOAP interface function **CreateTask()** are not shown in SMG Transfer Utility if you had it installed.
3. Call function **GetTaskStatus()** any time during execution by providing parameter with the task ID to retrieve job execution status, whether it is waiting in the queue, is running, failed, completed successfully or canceled .

   Call function **AbortTask()** to cancel a job any time after it is created by providing parameter with the task ID.

   Call function **GetTaskList ()** any time to get a list of all jobs or specifically a list of all running jobs depending on the parameters you supply when invoking the call.
4. Call function **GetResourceList()** any time to get info on which Media Engines are available and what jobs are executed by which Media Engine. Note it is viewable only. That means you cannot make any changes to it including redistributing Media Engine resource among different jobs because those are handled internally by SMG application as it is designed.

All above function calls are summarized in the following flowchart.


GetConfiguration()
SMG Configure data changed?
Yes
Reload()
No
CreateTask()
GetTaskStatus()
AbortTask()
GetTaskList()
END

2 SOAP Interface Functions

This chapter specifies SOAP interface provided by SMG Gateway Service. XML data shown in the following sample are not case-sensitive.

2.1 GetConfiguration

GetConfiguration(std::string& sConfig);

Function: Get configuration info which is defined in SMG Server.

Parameter:

[Out] sConfig –SMG configuration info (SMGConfiguration Sample xml format)

SMGConfiguration Sample xml format: *(**Bold** is for policy case)*

<?xml version="1.0" encoding="UTF-8"?>

<SMGConfiguration version="3.0">

<LicenseConfig IP="10.20.5.1" Port="37931"/>

<MPCServiceConfig Number="1">

<MPCService MPCServiceID="10.20.5.5_s" IPAddress="10.20.5.5" IPPort="30001" LogFile="C:\log\MPCService.log" FailOverFile="C:\log\MPCFailOver.xml" TempFileDir="D:\MediaData" CurTask="0" MaxTask="4" ShakeTime="40" QueryTaskTime="40" PDType="0" HostName="" InstallPath="" UserName="administrator" PassWord="cdci">

<DeviceList>

<Device DeviceName="10.20.1.11_ssapi"/>

<Device DeviceName="192.168.81.168_ftp"/>

</DeviceList>

</MPCService>

</MPCServiceConfig>

<PolicyConfiguration Number="1">

<**Policy Type**="2">

<**Manual Name**="Demo_ftp" Description="sd" TimeOut="600000" TaskKeepTime="40000" FailOverFile="">

<MediaEngineList>

<MediaEngine Name="10.20.5.5_s"/>

</MediaEngineList>

<MPCTASK TaskType="2" DeleteSourceFile="0" WithMetaData="0" DiscardTargetFileExt="0" ReplaceExistingFile="1">

<SourceMaterialList materialType="39" materialCount="1" PDVersion="18" VIXVersion="32" DeviceType="10" DeviceName="192.168.81.168_ftp">

<MPCMATERIALDEF materialName="" materialType="39" locationHost="192.168.81.168_ftp" materialLocationType="0" locationIPAddr="" locationPort="0" locationPath="" locationUser="" locationPassword="" locationCategory="" locationRes1="" locationRes2="0" ContinuousPick="0" Speed="0"/>

</SourceMaterialList>

<DestinationMaterialList materialType="11" materialCount="3" PDVersion="18" VIXVersion="32" DeviceName="10.20.1.11_ssapi" DeviceType="4">

<MPCMATERIALDEF materialName="" materialType="2" locationHost="10.20.1.11_ssapi" materialLocationType="0" locationIPAddr="" locationPort="0" locationPath="" locationUser="" locationPassword="" locationCategory="" locationRes1="" locationRes2="0" ContinuousPick="0" Speed="0"/>

<MPCMATERIALDEF materialName="" materialType="7" locationHost="10.20.1.11_ssapi" materialLocationType="0" locationIPAddr="" locationPort="0" locationPath="" locationUser="" locationPassword="" locationCategory="" locationRes1="" locationRes2="0" ContinuousPick="0" Speed="0"/>

<MPCMATERIALDEF materialName="" materialType="8" locationHost="" materialLocationType="2" locationIPAddr="10.20.1.11" locationPort="2001" locationPath="" locationUser="administrator" locationPassword="cdci" locationCategory="" locationRes1="" locationRes2="0" ContinuousPick="0" Speed="0"/>

</DestinationMaterialList>

</MPCTASK>

</Manual>

</Policy>

</PolicyConfiguration>

<AudioMuxAgent Number="0"/>

</SMGConfiguration>

Tag	Field Name	Value	Description	Comment
PolicyConfiguration	number	1	Policy list count	Defined policy numbers
Policy	Type	2	Policy type	1- manual policy
Manual	Name	Transfer with metadata	Policy name	Used for create task by policy

Remarks

Client can get SMG configuration info and parse configuration to the manual policy list.

If configuration is loaded successfully, return SOAP_OK.

Otherwise get error message from returned Soap structure.

2.2 Reload

Reload(std::string sCommand, std::string& s);

Function: Forced to reload configuration.

Parameter:

[In]sCommand – reload configuration command info.

[Out]sResult – reply message format.(SMGResult Sample xml format)

```
<?xml version="1.0" encoding="UTF-8"?>
<smgcfg command="Reload" version="1.0" clientid="client1" reconnectmpc="0" />
```

Tag	Field Name	Value	Description	Comment
Smgcfg	command	Reload	Client can reload configuration	Required
	clientid	client1	Unique Client id	Required. unique
	reconnectmpc	0(default)	Gateway service will reload configuration and reconnect all Media Engines if value is 1.	Reserved 0 – no 1 - reconnect all Media Engines

SMGResult Sample xml format

```
<?xml version="1.0" encoding="UTF-8"?>
<smgResult reply="Reload" return="TRUE" errorcode="0" errormsg="">
</smgResult>
```

errorcode	errormsg	memo
0xFE40	Failed to load SMGConfiguration.xml.	

Remark

Same as GetConfiguration.

2.3 CreateTask

CreateTask(std::string sCommand, std::string& sResult);

Function: Forced to run task immediately

Parameter:

[In]sCommand – create task command info.(SMGCreateTask Sample xml format)

[Out]sResult – reply message format.(SMGResult Sample xml format)

If MPCAPI can not be initialized like no license, no configuration, return soap error;

If task command info is invalid (see table), return soap error;

Else return SOAP_OK.

Remarks

Client can create a task based on Policy. For such tasks, client should only specify clientid(unique within clients), source clip name, and Policy name.

SMGCreateTask Sample xml format

Policy task:

```
<?xml version="1.0" encoding="UTF-8"?>
<smgtask command="CreateTask" version="1.0" clientid="Client1">
<task uid="" priority="1"/>
<policy name="mxfvixgen_local"/>
<source filename ="3d4e.mpg" />
</smgtask>
```

Smgtask	clientid	MediaMonitor	string to identify unique client. This filed should be unique.	Required, If string length>32 or <=0, return FALSE
Policy	name	Demo_ftp	Unique policy name which is defined in SMG	Required for Policy based task.
Source	filename	3d4e.mpg	target file name or folder name	Whether target is file or folder is depended on Preset.
Additions	reserved2	0		Reserved for SMG internal protocol

SMGResult Sample xml format:

```
<?xml version="1.0" encoding="UTF-8"?>
<smgResult reply="CreateTask" return="TRUE" errorcode="0" errormsg="" id="">
</smgResult>
```

Task should be executed immediately. SMG will check whether it has resource to execute the task before it sends acknowledgement to client. And SMG should return error if any precondition is not satisfied, or task uid is already present. Error code should be defined for different error.

Note: SMG Gateway Service will not check whether source and destination device can be accessed, whether clip exists before sending responses immediately. The status and result must be queried by client.

Task-uid is supplied by client. Client should ensure that it is unique. Task-id is returned by CreateTask. SMG will ensure that it is unique.

Both of them can be used to AbortTask or GetTaskStatus.

Task-uid can be empty. In the case, only Task-id is used.

2.4 GetTaskStatus

GetTaskStatus(std::string sCommand, std::string &sStatu);

Function: Query task status

Parameter:

[In]sCommand – get task status command info.(SMGGetTaskStatus Sample xml format)

[Out]sStatus – status info (SmgtaskInfo Sample xml format)

SMGGetTaskStatus Sample xml format:

<?xml version="1.0" encoding="UTF-8"?>

<smgtask **command**="GetTaskStatus" version="1.0" **clientid**="Client1">

<**task** uid=”FDE252190EB440aeBFE7FC4A71DB5393” id=”” />

</smgtask>

SmgtaskInfo Sample xml format:

<?xml version="1.0" encoding="UTF-8"?>

<smgtask **reply**="GetTaskStatus" version="1.0" **clientid**="Client1" >

<tasklog id=”” **status**=”RUN” **progress**="50" mpcapiid="" **mpcsrvip**="" source_device="" target_device="" source_file="" target_file="" source_file_type="" target_file_type="" source_file_size="" read_size="" write_size="" start_time="" end_time="" MD5code=" " uid="FDE252190EB440aeBFE7FC4A71DB5393" **errorcode**="" **errormsg**="" manifest=""/>

</smgtask>

Tag	Field Name	Value	Description	Comment
tasklog	uid	FDE252190EB440aeBFE7FC4A71DB5393	UUID, must be unique in task list	
	status	RUN	string of status	WAITDISPATCH – wait dispatch(task is queued and wait to be dispatched) WAITRUN – wait run(task has been dispatched to MPCService) RUN – is running COMPLETE – is ended successfully ERROR – error when run task ABORT – task is cancelled WAITABORT – task is about to be

				aborted DISCONNECT – unknown status due to lost connection with Media Engine
	progress	50	Running progress percent	if status is RUN, progress is available. but if source clip is streaming, progress=”unknown”
	mpcsrvip	192.168.81.164	Media Engine ip address	The task is running on this media engine machine
	errorcode		Error code defined in SMG	If status= ERROR, errorcode and error message are available
	errormsg		Error message defined in SMG	Same as errorcode

Remarks

Client can query status of a task by task uid. SMG should have a unified status chart for all types of task. Status includes clientid, task uid, task status, task progress in percentage.

If MPCAPI cannot be initialized like no license, no configuration, return SOAP message code.

If it cannot find task uid in task list created by clientid or SMGGetTaskStatus XML format is invalid, only errorcode and errormsg can be got from SMGTaskInfo XML.

Return SOAP_OK.

No callback for task status notification in current version.

2.5 AbortTask

AbortTask(std::string sCommand, std::string& sResult);

Function: Forced to cancel task immediately. If task is running, this command will mark task as WaitAbort and return. Client need get task status later till task is Aborted.

Parameter:

[In]sCommand – abort task command info.(SMGAbortTask Sample xml format)

[Out]sResult – reply message format.(Refer to SMGResult Sample xml format)

If MPCAPI cannot be initialized like no license, no configuration, return FALSE;

If cannot find task uid in task list created by clientid, return FALSE;

If task status cannot be aborted due to ended with ABORT, COMPLETE, ERROR, DISCONNECT return FALSE; Else return TRUE.

SMGAbortTask Sample xml format:

```
<?xml version="1.0" encoding="UTF-8"?>
<smgtask command="AbortTask" version="1.0" clientid="Client1">
<task uid=”FDE252190EB440aeBFE7FC4A71DB5393” id=””/>
</smgtask >
```

Remark

Client can cancel an in-processing task. SMG will send acknowledgement before the task is cancelled. Client need query task status whether it has actually been cancelled.

2.6 GetTaskList

GetTaskList(std::string sCommand,std::string& allTaskInfo);

Function: query task list info

Parameter:

[In]sCommand – get task list command info(SMGGetTaskList Sample xml format)

[Out]allTaskInfo – task list info(SMGAllTaskInfo Sample xml format)

SMGGetTaskList Sample xml format:

<?xml version="1.0" encoding="UTF-8"?>

<smgtask **command**="GetTaskList" version="1.0" clientid="Client1">

<tasklist **filterstatus**=""/>

</smgtask>

Tag	Field Name	Value	Description	Comment
tasklist	filterstatus	RUN	Filter string of status	“” – get all tasks “RUN” – get running tasks

SMGAllTaskInfo Sample xml format:

<?xml version="1.0" encoding="UTF-8"?>

<smgtask reply="GetTaskList" version="1.0" clientid="eCWM">

<tasklist filterstatus="" **taskcount**="2">

<task id=" " status="RUN" progress="50" mpcapiid="" mpcsrvip="192.168.81.164" source_device="" target_device="" source_file="" target_file="" source_file_type="" target_file_type="" source_file_size="" read_size="" write_size="" start_time="" end_time="" uid=" FDE252190EB440aeBFE7FC4A71DB5393" errorcode="" errormsg=""/>

<task id=" " status="COMPLETE" progress="100" mpcapiid="" mpcsrvip="192.168.81.167" source_device="" target_device="" source_file="" target_file="" source_file_type="" target_file_type="" source_file_size="" read_size="" write_size="" start_time="" end_time="" MD5code=" " uid=" CFG12345EB440aeBFE7FC4A71DB5394" errorcode="" errormsg="" manifest=""/>

</tasklist>

</smgtask>

Tag	Field Name	Value	Description	Comment
smgtask	taskcount	2	Number of matching status	If no task can be matched, result will also return TRUE.

Remarks

Client can query the task queue that is currently running on all Media Engines. The task life cycle is managed by MPCAPI.

When task is ended, the task record will not be freed right away. It is still kept in memory for a fixed period of time which is set as default in MPCAPI. Client must query task status during its lifetime.

If MPCAPI cannot be initialized like no license, no configuration, return soap error message; else return SOAP_OK.

2.7 GetResourceList

GetResourceList(std::string& allresInfo);

Function: query Media Engine resource info

Parameter:

[Out]allresInfo – resource list info(SMGAllResInfo Sample xml format)

SMGAllResInfo Sample xml format:

<?xml version="1.0" encoding="UTF-8"?>

<smgtask reply="GetResourceList" version="1.0" clientid="Client1">

<resource **status**=”free”>

<license **task_count**="16" />

<mpc name="MPC1" ip="192.168.81.164" port="30001" status="ONLINE" **max_task**="4" **cur_task**="0" />

<mpc name="MPC2" ip="192.168.81.167" port="" status="OFFLINE" **max_task**="4" **cur_task**="4" />

</resource >

</smgtask>

Tag	Field Name	Value	Description	Comment
smgtask				
resource	status	“free” / “busy”	SMG group status	If any mpc is not busy. And task count is less than license task_count. This value is free, else busy.
license	task_count	16	The limitation of concurrent task number in license	The value is used for client to judge whether SMG is busy or not.
mpc	name	MPC1	MediaEngine name	
	ip	192.168.81.164	Media Engine server ip address	
	port	30001	TCP port connected with MPCAPI	
	status	ONLINE	string of Media Engine status	ONLINE – connected OFFLINE – lost connection
	max_task	4	Max sessions defined in Configuration	In Carbon Coder case max session should be 1 even if its platform can support high performance.
	cur_task	1	Current running task number	

Remarks

Client can query resource list and determine whether there is available/used/total resource. It can also get resource's status.

For concurrent tasks SMG can execute immediately, it was also controlled by License.

If cur_task >= max_task, MPCAPI will not dispatch task but wait;

If total cur_task number >= License_task_count, MPCAPI will also not dispatch task but wait.

Client can send command based on this logic.

If MPCAPI cannot be initialized like no license, no configuration, return Soap error message; else return SOAP_OK.

2.8 SetTaskPriority(TBD)

SetTaskPriority(std::string sCommand, std::string& result);

Function: Forced to set task's priority.

Parameter:

[In]sCommand – Change task priority command info(SMGSetTaskPriority sample xml format)

[Out]sResult – reply message format.(SMGResult Sample xml format)

If MPCAPI cannot be initialized like no license, no configuration, return FALSE;

If cannot find task uid in task list created by clientid, return FALSE;

If task status is not "WAITDISPATCH" or "WAITRUN", return FALSE;

Else return TRUE.

SMGSetTaskPriority Sample xml format:

```
<?xml version="1.0" encoding="UTF-8"?>
<smgtask command="SetTaskPriority" version="1.0" clientid="Client1">
<task uid="FDE252190EB440aeBFE7FC4A71DB5393" id="" priority="0" />
</smgtask >
```

Tag	Field Name	Value	Description	Comment
task	priority	0	The priority of task	0 – High 1- Normal 2- Low ... 100 Lowest

3 Error Codes

ErrorCode will output from MPCService and MPCAPI.

Detailed descriptions are given in MPC Error Code Definition document.

1. CreateTask

Error code	Error message
0xFE71	Failed to load create task XML with invalid format.
0xFE72	Failed to load create task XML with root element.
0xFE73	Failed to load create task XML with source element.
0xFE74	Failed to load create task XML with target element.
0xFE75	Failed to load create task XML with UUID element.
0xFE76	Failed to load create task XML with Additions element.
0xFE77	Failed to load create task XML with task element.
0xFE81	Failed to create task because same client task UUID has existed.
0xFE82	Failed to create task because unknown preset GUID is used.
0xFE87	Failed to create task by XML because autodetect is not supported.
0xFE90	Invalid client task UUID.
0xFE40	Failed to load SMGConfiguration.xml.
0xFE30	Failed to Initialize.
0xFE31	Failed to load license.
0xFE25	This conversion with preset is not supported without pro-conversion.

2. GetTaskStatus

Error code	Error message
0xFE7A	Failed to parse get status command XML.
0xFE90	No task found by this ID.
0xFE91	Cannot get task by this ID.
0xFE85	Invalid client task UUID.
0xFE86	Unrecognized client task UUID.
0xFE25	This conversion with preset is not supported without pro-conversion.
0xFE2A	No online MPCService available.
0xFE2B	Failed to send task message to MPCService.
0xFEE1	Failed to dispatch task because running task count exceeds license setting.
0xFEE2	Failed to dispatch task because running task count exceeds MPC group upper limit.
0xFEE3	No free MPCService available.

0x00100000	Failed to init transcoder.
0x00100001	This operation is not supported in this version.
0x00100002	An unknown error occurred.
0x00100003	A version conflict occurred.
0x00100100	General error.
0x00100101	Invalid job ID.
0x00100102	Failed to create job.
0x00100103	Failed to execute job.
0x00100104	Failed to query job info.
0x00100105	Failed to stop job.
0x00100000 + x	Carbon Coder error code and message. Please refer to “Carbon User Manual.pdf”.

3. AbortTask

Error code	Error message
0xFE78	Failed to parse abort command XML.
0xFE51	Failed to abort task because task is not found.
0xFE52	Failed to abort task because task is ended.
0xFE90	Invalid client task UUID.
0xFE91	Unrecognized client task UUID.

4. Reload

Error code	Error message
0xFE40	Failed to load SMGConfiguration.xml
0xFE30	Failed to Initialize.