Dialogic®

Dialogic® PowerMedia™ XMS RESTful Management API

Developer's Guide

January 2013 05-2711-001

www.dialogic.com

Copyright and Legal Notice

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Inc. and its affiliates or subsidiaries ("Dialogic"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in certain safety-affecting situations. Please see http://www.dialogic.com/company/terms-of-use.aspx for more details.

Due to differing national regulations and approval requirements, certain Dialogic products may be suitable for use only in specific countries, and thus may not function properly in other countries. You are responsible for ensuring that your use of such products occurs only in the countries where such use is suitable. For information on specific products, contact Dialogic Inc. at the address indicated below or on the web at www.dialogic.com.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by or on web pages maintained by Dialogic may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic's legal department at 6700 de la Cote-de-Liesse Road, Suite 100, Borough of Saint-Laurent, Montreal, Quebec, Canada H4T 2B5. **Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements**.

Dialogic, Dialogic Pro, Dialogic Blue, Veraz, Brooktrout, Diva, BorderNet, PowerMedia, ControlSwitch, I-Gate, Mobile Experience Matters, Network Fuel, Video is the New Voice, Making Innovation Thrive, Diastar, Cantata, TruFax, SwitchKit, Eiconcard, NMS Communications, SIPcontrol, Exnet, EXS, Vision, inCloud9, NaturalAccess and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Inc. and its affiliates or subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 6700 de la Cote-de-Liesse Road, Suite 100, Borough of Saint-Laurent, Montreal, Quebec, Canada H4T 2B5. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

The names of actual companies and products mentioned herein are the trademarks of their respective owners.

This document discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.

Table of Contents

Revision History

Revision	Release Date	Notes
05-2711-001	January 2013	Updates to support PowerMedia XMS Release 2.0. RESTful Management API Resources: • Added the RESTful Management API Resources.
05-2711-001-01	October 2012	Initial release of this document.
Last modified: January 2013		

Refer to www.dialogic.com for product updates and for information about support policies, warranty information, and service offerings.

1. Welcome

The Dialogic® PowerMedia™ Extended Media Server (also referred to herein as "PowerMedia XMS" or "XMS") RESTful Management API provides an alternate way of performing system management tasks. Although the PowerMedia XMS is usually configured and managed through an interactive web-based GUI, there may be circumstances where system management could be done in a more automated or distributed manner.

This Developer's Guide provides instructions for using the RESTful Management API.

2. Overview

The RESTful Management API is used to control the OA&M functions of PowerMedia XMS. Normally, this is done through a GUI, running on a client web browser. However, there may be a need to customize system management functionality, or merge it in with other system management tools in use. The RESTful Management API can be used to accomplish this.

The RESTful Management API uses HTTP or secure HTTPS as its transport so that information about the PowerMedia XMS server configuration cannot be intercepted. In the RESTful Management API, the four HTTP methods are translated to the actions shown in the following table.

HTTP Method	Action
POST	Add a new resource
PUT	Modify an existing resource
GET	Return information about a resource
DELETE	Remove a resource

Request and response payloads must be specified using JavaScript Object Notation (JSON). JSON is a lightweight data-interchange format based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including Java, JavaScript, Perl, and Python.

A reference of RESTful Management resources is found in RESTful Management API Resources. Only a few example resources and JSON payloads will be mentioned in this guide.

The RESTful Management API is active once the PowerMedia XMS is installed and running.

3. Firefox RESTClient Utility

While the RESTful Management API is intended to be used as part of an overall management system written by the customer, there is convenient way to exercise the API and become familiar with its operation. The Firefox web browser (http://www.mozilla.org) supplies a free extension, RESTClient, which allows RESTful methods to be generated and sent to a RESTful service. It is similar to the XMSTool used with the call and media control RESTful API.

Accessing RESTClient

Proceed as follows to access RESTClient:

1. From an up-to-date Firefox browser (Version 18.0 as of January 2013), use the main Firefox pull-down menu and select **Add-ons**.
2. Enter **restclient** in the search box at the upper right corner of the page.
   An entry for **RESTClient, a debugger for RESTful web services** will be observed under **Available Add-ons**.
3. Click on **Install**.
4. Restart Firefox when the installation is complete. The RESTClient will appear under the Extensions tab in Add-ons.

4. Configuring and Using the RESTClient Utility

Before the RESTClient is started, a security exception must be put in place for the HTTPS connection that will be used to PowerMedia XMS. Use a regular Firefox browser window, (not the RESTClient) and go to **https://<xms_ip_address>:10443**.

A security exception will be generated by the non-verified security certificate supplied with PowerMedia XMS for the RESTful Management API. Handle the exception with: Understand the Risks/Add Exception/Confirm Security Exception. Allow the exception will remain permanently stored. For more information, refer to the CentOS HTTPS Setup for Console Use section in the *Dialogic® PowerMedia™ XMS Quick Start Guide*.

Proceed as follows to start RESTClient:

1. From the main Firefox pull-down menu, select **Web Developer**.
2. Click on **RESTClient**.
3. Select **Headers** and **Custom Header**.
4. Enter:
   Name: Content-Type, Value: application/json
   Name: Accept, Value: application/json

Note: Two Request Headers must be entered.

Basic Authentication should also be activated. Under Authentication/Basic Authentication, set:

1. Enter User Name: **admin**
   Enter Password: **admin**
   Remember Me – checked off
2. Select **OK**.

The remainder of this section gives examples for each supported HTTP method. However, refer to RESTful Management API Resources for details on using the resource and method.

Using GET with the RESTClient

1. Select **GET** from the Method pull-down menu.
2. Enter the HTTPS URL for the resource desired.
   The format used here is **https://<xms_ip_address>:10443/resource**.
   For example, to look at the domain name server setting (DNS) for the XMS system at 172.31.0.2, the URL would be **https://172.31.0.2:10443/network/dns**.
3. View the Response Header and Response Body by clicking their respective tabs.

The following example shows the JSON representation of the DNS requested above:

```
{
        "hostname" : "novalocal",
        "search" : "novalocal",
        "nameserver" : [
                {
                     "addr" : "172.31.0.3"
                }
        ]
}
```

Using PUT with the RESTClient

1. Select **PUT** from the Method pull-down.
2. Enter the HTTPS URL for the resource desired.
   The format is **https://<xms_ip_address>:10443/resource**.
   For example, to set the SIP call routing to be used, the URL would be **https://172.31.0.2:10443/routing**.
   The JSON describing the desired routes would be entered in the Request Body field.

The following example shows how to set all routes to their original values:

```
{
        "routes" : [
                {
                        "pattern" : "^sip:annc.*",
                        "application" : "NETANN"
                },
                {
                        "pattern" : "^sip:conf=.*",
                        "application" : "NETANN"
                },
                {
                        "pattern" : "^sip:dialog.*",
                        "application" : "VXML"
                },
                {
                        "pattern" : "^sip:.*",
                        "application" : "app"
                }
         ]
}
```

Requests can be entered into the Body text field, copied and pasted into the field or read out of a file.

Using POST with the RESTClient

1. Select **POST** from the Method pull-down menu.
2. Enter the HTTPS URL for the resource desired.
   The format is **https://<xms_ip_address>:10443/resource**.
   For example, to create a system backup, the URL would be **https://172.31.0.2:10443/system/backup**.

The following example shows what would be returned in the Response Body:

```
{
        "system_backup" : [
                  {
                       "resources" : [
                  {
                                    "id" : "xmsbackup-20121011-161653.tar.gz",
                                    "uri" : "\/system\/backup\/xmsbackup-20121011-161653.tar.gz"
                  }
                        ]
         }
}
```

In addition, the backup file `/etc/xms/backup/xmsbackup-20121011-161653.tar.gz` would be created.

Using DELETE with the RESTClient

1. Select **DELETE** from the Method pull-down menu.
2. Enter the HTTPS URL for the resource desired.
   The format is **https://<xms_ip_address>:10443/resource**.
   For example, to delete the media file `/var/lib/xms/media/en-US/vxml/recorded/record-120921-085752-0000.wav`, the URL would be **https://172.31.0.2:10443/media/vxml/recorded/record-120921-085752-0000.wav**.

Note: The root media directory is `/var/lib/xms/media/en-US/` and does not need to be specified.

5. RESTful Management API Resources

The RESTful Management API supports JSON data format. The format is specified using standard mime-types. The client should use the Accept header to specify the format that it wishes to receive. The Content-Type header specifies the mime-type of the request's or response's content.

The RESTful Management API can be accessed in two ways:

- Locally over HTTP
  Use the base URL http://127.0.0.1:10080/
- Externally over HTTPS
  The https interface is provided by the lighttpd service acting as proxy.
  Use https://<ipaddress>:10443/
  The https service uses http basic authentication and defines a single user "admin" with the password "admin".

PowerMedia XMS has two modes of operation, Native and MSML. The resources provided by the RESTful Management API will differ between these modes. For example, when in MSML mode, the Routing, Tones, MRCP Client and VXML resources are not available.

The table below lists all available resources, their sub-resources, valid HTTP methods that may be used with them and XMS modes for which they are valid.

Clicking on the resource or sub-resource will:

- Provide its definition
- Provide valid values for the parameters that can be set
- Define how each valid method affects the resource
- Give an example of a request and a response payload

Resource	Sub-Resource	HTTP Methods Supported	PowerMedia XMS Modes Supported
/system		GET, PUT	Native, MSML
	/info	GET	Native, MSML
	/time	GET, PUT	Native, MSML
	/backup	GET, POST, DELETE	Native, MSML
	/restore	GET, PUT	Native, MSML

Resource	Sub-Resource	HTTP Methods Supported	PowerMedia XMS Modes Supported
	/upgrade	GET, PUT, POST, DELETE	Native, MSML
	/nfsmount	GET, PUT	Native, MSML
/sip		GET, PUT	Native, MSML
/rtp		GET, PUT	Native, MSML
/trust		GET, PUT	Native, MSML
/services		GET, PUT	Native, MSML
/license		GET, PUT, POST, DELETE	Native, MSML
/codecs		GET, PUT	Native, MSML
/routing		GET, PUT	Native
/msml		GET, PUT	MSML
/media		GET, POST, DELETE	Native, MSML
/tones		GET, PUT	Native, MSML
/network		GET	Native, MSML

Resource	Sub-Resource	HTTP Methods Supported	PowerMedia XMS Modes Supported
	/interface	GET, PUT	Native, MSML
	/dns	GET, PUT	Native, MSML
/mrcpclient		GET	Native
	/global	GET, PUT	Native
	/speechserver1	GET, PUT	Native
	/speechserver2	GET, PUT	Native
/vxml		GET	Native
	/vxmlinterpreter	GET, PUT	Native
/xmsrest		GET	Native
	/general	GET, PUT	Native
	/trusted	GET, PUT	Native

6. List of Available Resources

/system

Resource URI

```
/system
```

Resources

Resource	Description
info	Gets system information resource
network	Gets network configuration resource
time	Gets time and date configuration resource

HTTP GET

Retrieves all available system resources.

```
GET /system
```

Method	Response Payload

JSON	{
	  "state" : "running",
	  "version" : "1.0",
	  "resources" : [
	  {
	  "uri" : "\/system\/info"
	  },
	  {
	  "uri" : "\/system\/network"
	  },
	  {
	  "uri" : "\/system\/time"
	  }
	  ]
	}

HTTP PUT

Restarts or shutdowns the server.

```
PUT /system
```

Valid "state" settings:

- shutdown
- restart
- running (readonly)

Method	Request Payload	Response Payload

JSON	<pre>{
	  "state" : "shutdown"
	}</pre>
		<pre>{
		  "state" : "shutdown",
		  "resources" : [
		  {
		  "uri" : "\/system\/info"
		  },
		  {
		  "uri" : "\/system\/network"
		  },
		  {
		  "uri" : "\/system\/time"
		  }
		  ]
		}</pre>

/system/info

Resource URI

```
/system/info
```

HTTP GET

Retrieves information for a single resource.

```
GET /system/info
```

Method	Response Payload

JSON	<pre>{
	  "os_release" : "Fedora release 14 (Laughlin) ",
	  "os_version" : "Linux version 2.6.35.14-96.fc14.i686.PAE (mockbuild@x86-01.phx2.fedoraproject.org) (gcc version 4.5.1 20100924 (Red Hat 4.5.1-4) (GCC)) #1 SMP Thu Sep 1 12:31:46 UTC 2011 ",
	  "uptime" : 3017268,
	  "load_avg" : [
	  {
	  "1" : 0.13,
	  "5" : 0.1,
	  "15" : 0.14
	  }
	  ],
	  "memory" : [
	  {
	  "total" : 4123224,
	  "used" : 3206668
	  }
	  ],
	  "disk" : [
	  {
	  "total" : 145947208,
	  "used" : 98588080
	  }
	  ]
	}</pre>

/system/time

Resource URI

```
/system/time
```

HTTP GET

Retrieves information for system time.

```
GET /system/time
```

Method	Response Payload

JSON	{
	"system_time": {
	"time": "Fri Oct 26 19:15:39 2012",
	"zone": "America/New_York",
	"utc": "no",
	"syncntp": "yes",
	"ntpserver": [
	{
	"addr": "0.centos.pool.ntp.org",
	"iburst": "false",
	"maxpoll": "10",
	"minpoll": "6"
	},
	{
	"addr": "1.centos.pool.ntp.org",
	"iburst": "false",
	"maxpoll": "10",
	"minpoll": "6"
	},
	{
	"addr": "2.centos.pool.ntp.org",
	"iburst": "false",
	"maxpoll": "10",
	"minpoll": "6"
	}
	]
	}
	}

HTTP PUT

Modifies system time.

```
PUT /system/time
```

Method	Request Payload	Response Payload

JSON	<pre>{
	"system time":
	{
	"zone": "America/xyz",
	"utc": "yes",
	"syncntp": "yes",
	}
	}</pre>
		<pre>{
		"system time":
		"time": "Fri Oct 26 19:15:39 2012",
		"zone": "America/xyz",
		"utc": "yes",
		"syncntp": "yes",
		"ntpserver": [
		{
		"addr": "0.centos.pool.ntp.org",
		"iburst": "false",
		"maxpoll": "10",
		"minpoll": "6"
		},
		{
		"addr": "1.centos.pool.ntp.org",
		"iburst": "false",
		"maxpoll": "10",
		"minpoll": "6"
		},
		{
		"addr": "2.centos.pool.ntp.org",
		"iburst": "false",
		"maxpoll": "10",
		"minpoll": "6"
		}
		]
		}
		}</pre>

/system/backup

Resource URI

```
/system/backup
```

HTTP GET

Retrieves all available backup files.

```
GET /system/backup
```

Method	Response Payload

JSON	{
	  "system_backup": {
	  "resources": [
	  {
	  "id": "xmsbackup-20120619-160125.tar.gz",
	  "uri": "/system/backup/xmsbackup-20120619-160125.tar.gz"
	  },
	  {
	  "id": "xmsbackup-20120620-120928.tar.gz",
	  "uri": "/system/backup/xmsbackup-20120620-120928.tar.gz"
	  },
	  {
	  "id": "xmsbackup-20120620-120935.tar.gz",
	  "uri": "/system/backup/xmsbackup-20120620-120935.tar.gz"
	  }
	  ]
	  }
	}

Single Instance

Resource URI

```
/system/backup/xmsbackup-20120619-160125.tar.gz
```

Download a single backup file.

```
GET /system/backup/xmsbackup-20120619-160125.tar.gz
```

Response

A binary data of the backup file.

HTTP POST

Creates a system backup file.

```
POST /system/backup
```

Method	Response Payload

JSON	{
	  "system_backup": {
	  "resources": [
	  {
	  "id": "xmsbackup-20120619-160125.tar.gz",
	  "uri": "/system/backup/xmsbackup-20120619-160125.tar.gz"
	  },
	  {
	  "id": "xmsbackup-20120620-120928.tar.gz",
	  "uri": "/system/backup/xmsbackup-20120620-120928.tar.gz"
	  },
	  {
	  "id": "xmsbackup-20120620-120935.tar.gz",
	  "uri": "/system/backup/xmsbackup-20120620-120935.tar.gz"
	  }
	  ]
	  }
	}

Uploads a system backup file.

Payload content-type="application/x-gzip"

```
POST /system/backup/mybackup.tar.gz
```

Method	Response Payload

JSON	{
	  "system_backup": {
	  "resources": [
	  {
	  "id": "xmsbackup-20120619-160125.tar.gz",
	  "uri": "/system/backup/xmsbackup-20120619-160125.tar.gz"
	  },
	  {
	  "id": "xmsbackup-20120620-120928.tar.gz",
	  "uri": "/system/backup/xmsbackup-20120620-120928.tar.gz"
	  },
	  {
	  "id": "mybackup.tar.gz",
	  "uri": "/system/backup/xmsbackup-20120620-120935.tar.gz"
	  }
	  ]
	  }
	}

HTTP DELETE

Deletes a system backup file.

```
DELETE /system/backup/mybackup.tar.gz
```

/system/restore

Resource URI

```
/system/restore
```

HTTP GET

Retrieves all available backup files for restore.

```
GET /system/restore
```

Method	Response Payload

JSON	{
	  "system_restore": {
	  "resources": [
	  {
	  "id": "xmsbackup-20120619-160125.tar.gz",
	  "uri": "/system/restore/xmsbackup-20120619-160125.tar.gz"
	  },
	  {
	  "id": "xmsbackup-20120620-120928.tar.gz",
	  "uri": "/system/restore/xmsbackup-20120620-120928.tar.gz"
	  },
	  {
	  "id": "xmsbackup-20120620-120935.tar.gz",
	  "uri": "/system/restore/xmsbackup-20120620-120935.tar.gz"
	  }
	  ]
	  }
	}

HTTP PUT

Restores system from a system backup file.

```
PUT /system/backup/xmsbackup-20120619-160125.tar.gz
```

/system/upgrade

Resource URI

```
/system/upgrade
```

HTTP GET

Retrieves all available upgrade files.

```
GET /system/upgrade
```

Method	Response Payload

JSON	<pre>{
	  "system_upgrade": {
	  "resources": [
	  {
	  "id": "dialogic_xms_trunk.3375.tgz",
	  "uri": "/system/upgrade/dialogic_xms_trunk.3375.tgz"
	  }
	  ]
	  }
	}</pre>

HTTP PUT

Upgrades system from a system upgrade file.

```
PUT /system/b/xmsbackup-20120619-160125.tar.gz
```

HTTP POST

Upload a system backup file.

Payload content-type="application/x-gzip"

```
POST /system/upgrade/dialogic_xms_trunk.3375.tgz
```

Method	Response Payload

JSON	<pre>{
	  "system_upgrade": {
	  "resources": [
	  {
	  "id": "dialogic_xms_trunk.3375.tgz",
	  "uri": "/system/upgrade/dialogic_xms_trunk.3375.tgz"
	  }
	  ]
	  }
	}</pre>

HTTP DELETE

Deletes a system upgrade file.

```
DELETE /system/upgrade/dialogic_xms_trunk.3375.tgz
```

/system/nfsmount

Resource URI

```
/system/nfsmount
```

HTTP GET

Retrieves all NFS mount points.

```
GET /system/nfsmount
```

Method	Response Payload

JSON	{
	  "nfsmounts": [
	    {
	      "device": "onyx2:/files/Builds/G4PP/",
	      "mount_point": "/mnt/jane",
	      "options": "defaults",
	      "status": "disable"
	    },
	    {
	      "device": "onyx:/files/Builds/G2MS/",
	      "mount_point": "/mnt/lingx2",
	      "options": "defaults",
	      "status": "disable"
	    }
	  ]
	}

HTTP PUT

Adds, deletes, or modifies NFS mount points.

```
PUT /system/nfsmount
```

Method	Request Payload	Response Payload

JSON	{
	  "nfsmounts": [
	    {
	      "device":
	"onyx2:/files/Builds/G4PP/",
	      "mount_point": "/mnt/jane2",
	      "options": "ro",
	    },
	    {
	      "device":
	"onyx:/files/Builds/G2MS/",
	      "mount point": "/mnt/lingx2",
	      "options": "defaults",
	    },
	    {
	      "device":
	"onyx:/files/Builds/new/",
	      "mount_point":
	"/mnt/onyx/new",
	      "options": "defaults",
	    }
	  ]
	}
		{
		  "nfsmounts": [
		    {
		      "device":
		"onyx2:/files/Builds/G4PP/",
		      "mount_point": "/mnt/jane2",
		      "options": "defaults",
		      "status": "enable"
		    },
		    {
		      "device":
		"onyx:/files/Builds/G2MS/",
		      "mount_point": "/mnt/lingx2",
		      "options": "defaults",
		      "status": "disable"
		    },
		    {
		      "device":
		"onyx2:/files/Builds/new/",
		      "mount point":
		"/mnt/onyx/new",
		      "options": "defaults",
		      "status": "enable"
		    }
		  ]
		}

/sip

Resource URI

```
/sip
```

HTTP GET

Retrieves the SIP interface settings.

```
GET /sip
```

Method	Response Payload

JSON	<pre>{
	  "Board_1" : {
	  "app_srv_access" : "no",
	  "sip_ipaddr" : "DEFAULT",
	  "sip_port" : "5060",
	  "transport" : "UDP"
	  }
	}</pre>

HTTP PUT

Modifies the SIP interface IP address, port number and transport. Enables an access control list that defines the hosts allowed to connect to the SIP interface.

```
PUT /sip
```

Valid "transport" settings:

- UDP
- UDP_TCP

Method	Request Payload	Response Payload

JSON	<pre>{
	  "Board_1" : {
	  "app_srv_access" : "no",
	  "sip ipaddr" : "DEFAULT",
	  "sip_port" : "5060",
	  "transport" : "UDP"
	  }
	}</pre>
		<pre>{
		  "Board_1" : {
		  "app_srv_access" : "no",
		  "sip ipaddr" : "DEFAULT",
		  "sip_port" : "5060",
		  "transport" : "UDP"
		  }
		}</pre>

/rtp

Resource URI

```
/rtp
```

HTTP GET

Retrieves a network interface name that is used by media engine RTP and a list of available interface names and IP addresses.

```
GET /rtp
```

Method	Response Payload

JSON	<pre>{
	  "ifname": "eth0",
	  "mode": "native",
	  "rtp_ipaddr_mode": "", /*msml mode only */
	  "rtp_ipaddr1": "", /*msml mode only */
	  "rtp_ipaddr2": "", /*msml mode only */
	  "rtp_ipaddr3": "", /*msml mode only */
	  "rtp_ipaddr4": "", /*msml mode only */
	  "ifnames": [
	    "eth0","eth1"
	  ],
	  "ifaddrs": [
	    "10.20.129.19","10.20.129.20"
	  ]
	}</pre>

HTTP PUT

```
PUT /rtp
```

Method	Request Payload	Response Payload

JSON	<pre>{
	  "ifname": "eth1",
	}</pre>
		<pre>{
		  "ifname": "eth0",
		  "mode": "native",
		  "rtp_ipaddr_mode": "", /*msml mode only */
		  "rtp_ipaddr1": "", /*msml mode only */
		  "rtp_ipaddr2": "", /*msml mode only */
		  "rtp_ipaddr3": "", /*msml mode only */
		  "rtp_ipaddr4": "", /*msml mode only */
		  "ifnames": [
		    "eth0","eth1"
		  ],
		  "ifaddrs": [
		    "10.20.129.19","10.20.129.20"
		  ]
		}</pre>

/trust

Resource URI

```
/trust
```

HTTP GET

Sets and retrieves the trusted hosts that can communicate via the SIP port.

```
GET /trust
```

Method	Response Payload

JSON	<pre>{
	"trusted_addresses" : [
	"192.168.195.12",
	"192.168.195.204"
	]
	}</pre>

HTTP PUT

Modifies the trusted hosts configuration.

```
PUT /trust
```

Method	Request Payload	Response Payload

JSON	<pre>{
	"trusted_addresses" : [
	"192.168.195.12",
	"192.168.195.204"
	]
	}</pre>
		<pre>{
		"trusted_addresses" : [
		"192.168.195.12",
		"192.168.195.204"
		]
		}</pre>

/httpclient

Resource URI

```
/httpclient
```

HTTP GET

Retrieves the HTTP client configuration.

```
GET /httpclient
```

Method	Response Payload

JSON	<pre>{
	"cache": "no",
	"max_stale": "33",
	"max_age": "500"
	}</pre>

HTTP PUT

Configures the HTTP client resource.

```
PUT /httpclient
```

Method	Request Payload	Response Payload

JSON	<pre>{
	"cache": "yes",
	"max_stale": "0",
	"max_age": "30"
	}</pre>
		<pre>{
		"cache": "yes",
		"max_stale": "0",
		"max_age": "30"
		}</pre>

/services

Resource URI

```
/services
```

HTTP GET

Retrieves all available system services and the overall run state of the system services.

Valid "state" values:

- STOPPED
- STARTING
- RUNNING
- STOPPING
- FAILED

Valid "mode" values:

- native
- msml

```
GET /services
```

Method	Response Payload

JSON	{
	  "mode":"native",
	  "state":"RUNNING",
	  "restart_required":"no",
	  "services" : [
	  {
	  "id" : "appmanager",
	  "uri" : "\/services\/appmanager"
	  },
	  {
	  "id" : "broker",
	  "uri" : "\/services\/broker"
	  },
	  {
	  "id" : "xmserver",
	  "uri" : "\/services\/xmserver"
	  },
	  {
	  "id" : "restapi",
	  "uri" : "\/services\/restapi"
	  },
	  {
	  "id" : "mediasubsystem",
	  "uri" : "\/services\/mediasubsystem"
	  }
	  ]
	}

Single Instance

Resource URI

```
/services/appmanger
/services/broker
/services/xmserver
/services/restapi
/services/msmlserver
/services/mediasubsystem
```

Retrieves information for a single service resource.

```
GET /services/broker
```

Method	Response Payload

JSON	<pre>{
	  "service" : {
	  "id" : "appmanager",
	  "state" : "RUNNING",
	  "description" : "The appmanager process"
	  }
	}</pre>

HTTP PUT

Modifies the system services overall run state and operation mode.

Valid "state" settings:

- RUNNING
- STOPPED

Valid "mode" settings:

- native
- msml

Note: The "restart_required" parameter is read-only and is an indication that any change requires a services restart.

Native Mode

```
PUT /services
```

Method	Request Payload	Response Payload

JSON	<pre>{
	  "mode" : "native",
	  "state" : "STOPPED",
	}</pre>
		<pre>{
		  "mode":"native",
		  "state":"STOPPING",
		  "restart_required":"no",
		  "services" : [
		  {
		  "id" : "appmanager",
		  "uri" :
		"\/services\/appmanager"
		  },
		  {
		  "id" : "broker",
		  "uri" : "\/services\/broker"
		  },
		  {
		  "id" : "xmserver",
		  "uri" :
		"\/services\/xmserver"
		  },
		  {
		  "id" : "restapi",
		  "uri" : "\/services\/restapi"
		  },
		  {
		  "id" : "mediasubsystem",
		  "uri" :
		"\/services\/mediasubsystem"
		  },
		  ]
		}</pre>

MSML Mode

```
PUT /services
```

Method	Request Payload	Response Payload

JSON	<pre>{
	  "mode" : "msml",
	  "state" : "STOPPED",
	}</pre>
		<pre>{
		  "mode":"msml",
		  "state":"STOPPING",
		  "restart_required":"no",
		  "services" : [
		  {
		  "id" : "msmlserver",
		  "uri" :
		"\/services\/msmlserver"
		  },
		  {
		  "id" : "mediasubsystem",
		  "uri" :
		"\/services\/mediasubsystem"
		  },
		  ]
		}</pre>

/license

Resource URI

```
/license
```

HTTP GET

Retrieves all available licenses and the current licensing mode of the system services.

```
GET /license
```

Method	Response Payload

JSON	{
	  "licenses" : {
	  "resources" : [
	  {
	  "id" : "default.lic",
	  "uri" : uri="\/license\/default.lic"
	  },
	  {
	  "id" : "temp.lic" ,
	  "uri" : uri="\/license\/temp.lic"
	  },
	  {
	  "id" : "production.lic",
	  "uri" : uri="\/license\/production.lic"
	  }
	  ],
	  "features" : [
	  {
	  "name" : "amr_audio_codec",
	  "value" : 0
	  },
	  {
	  "name" : "basic_audio",
	  "value" : 200
	  },
	  {
	  "name" : "hd_audio_codec",
	  "value" : 200
	  },
	  {
	  "name" : "lbr_audio_codec",
	  "value" : 0
	  },
	  {
	  "name" : "video",
	  "value" : 100
	  }
	  ]
	  }
	}

Single Instance

Retrieves information for a single license resource.

```
GET /license/production.lic
```

Method	Response Payload

JSON	{
	  "license" : {
	  "id" : "production.lic",
	  "type" : "permanent",
	  "expires" : "never",
	  "status" : "active",
	  "features" : [
	  {
	  "name" : "amr_audio_codec",
	  "value" : 0
	  },
	  {
	  "name" : "basic audio",
	  "value" : 200
	  },
	  {
	  "name" : "hd_audio_codec",
	  "value" : 200
	  },
	  {
	  "name" : "lbr_audio_codec",
	  "value" : 0
	  },
	  {
	  "name" : "video",
	  "value" : 100
	  }
	  ]
	  }
	}

HTTP PUT

Modifies an individual license status.

Valid "status" values:

- active
- inactive

```
PUT /license/production.lic
```

Method	Request Payload	Response Payload

JSON	<pre>{
	   "status" : "active"
	}</pre>
		<pre>{
		  "license" : {
		    "id" : "production.lic",
		    "type" : "permanent",
		    "expires" : "never",
		    "status" : "active",
		    "features" : [
		      {
		        "name" : "amr_audio_codec",
		        "value" : 0
		      },
		      {
		        "name" : "basic_audio",
		        "value" : 200
		      },
		      {
		        "name" : "hd_audio_codec",
		        "value" : 200
		      },
		      {
		        "name" : "lbr_audio_codec",
		        "value" : 0
		      },
		      {
		        "name" : "video",
		        "value" : 100
		      }
		    ]
		  }
		}</pre>

HTTP POST

Uploads a new license. On completion, the license will be in an inactive state. Use PUT to activate.

```
POST /license/<production-new.lic>
```

Method	Request Payload	Response Payload

JSON	The request payload is the raw license file data. The mime type must be text/plain.
	`Content-Type: text/plain`
		<pre>{
		  "license" : {
		    "id" : "production-new.lic",
		    "type" : "Permanent",
		    "expires" : "never",
		    "status" : "inactive",
		    "features" : [
		      {
		        "name" : "amr_audio_codec",
		        "value" : 0
		      },
		      {
		        "name" : "basic_audio",
		        "value" : 200
		      },
		      {
		        "name" : "hd_audio_codec",
		        "value" : 200
		      },
		      {
		        "name" : "lbr_audio_codec",
		        "value" : 0
		      },
		      {
		        "name" : "video",
		        "value" : 100
		      }
		    ]
		  }
		}</pre>

HTTP DELETE

Deletes a license. License must be inactive.

```
DELETE /license/production.lic
```

/codecs

Resource URI

```
/codecs
```

HTTP GET

Retrieves the audio and video codec settings. The codecs are returned in priority order.

```
GET /codecs
```

Method	Response Payload

JSON	<pre>{
	  "audio_codecs" : [
	  {
	    "amr-wb" : {
	      "enabled" : "yes"
	    }
	  },
	  {
	    "g723" : {
	      "enabled" : "yes"
	    }
	  },
	  {
	    "amr-nb" : {</pre>

Method	Response Payload


```
      "enabled" : "yes"
    }
  },
  {
    "g729" : {
      "enabled" : "yes"
    }
  },
  {
    "pcmu" : {
       "enabled" : "yes"
    }
  },
  {
    "pcma" : {
      "enabled" : "yes"
    }
  },
  {
    "g722" : {
      "enabled" : "yes"
    }
  },
  {
    "g726" : {
      "enabled" : "yes"
    }
  }
],
"video_codecs" : [
  {
     "h264" : {
       "enabled" : "yes",
       "bitrate" : "768000",
       "fps" : "25",
       "level" : "22",
       "profile" : "66",
       "sample_rate" : "0",
       "size" : "vga"
     }
  },
  {
    "h263" : {
       "enabled" : "yes",
       "bitrate" : "384000",
       "fps" : "30",
       "level" : "30",
       "profile" : "0",
       "sample_rate" : "0",
       "size" : "cif"
    }
  }
]
}
```


HTTP PUT

Modifies the supported codec configuration.

```
PUT /codecs
```

The following tables provide the valid settings for an XMS system running.

codec	level	size	fps	bps
h264	3.1	vga	30	2000000
h264	3.1	vga	30	768000
h264	3	vga	25	768000
h264	2.2	vga	25	768000
h264	2.1	cif	30	384000
h264	2	cif	30	384000
h264	1.3	cif	30	384000
h264	1.2	cif	15	384000
h264	1.1	qcif	30	192000
h264	1b	qcif	15	128000
h264	1	qcif	15	42000

codec	level	size	fps	bps
mp4v-es	3	cif	30	384000
mp4v-es	2	cif	15	128000
mp4v-es	1	qcif	15	64000
mp4v-es	0	qcif	15	42000

codec	level	size	fps	bps
h263	30	cif	30	384000
h263	30	cif	15	384000
h263	30	cif	10	384000
h263	30	qcif	30	384000
h263	30	qcif	15	384000

codec	level	size	fps	bps
h263	20	cif	30	128000
h263	20	cif	15	128000
h263	20	cif	10	128000
h263	20	qcif	30	128000
h263	20	qcif	15	128000

codec	level	size	fps	bps
h263-1998	30	cif	30	384000
h263-1998	30	cif	15	384000
h263-1998	30	cif	10	384000
h263-1998	30	qcif	30	384000
h263-1998	30	qcif	15	384000
h263-1998	20	cif	30	128000
h263-1998	20	cif	15	128000
h263-1998	20	cif	10	128000
h263-1998	20	qcif	30	128000
h263-1998	20	qcif	15	128000

Method	Request Payload	Response Payload

JSON	{
	"audio_codecs" : [
	{
	"amr-wb" : {
	"enabled" : "yes"
	}
	},
	{
	"g723" : {
	"enabled" : "yes"
	}
	},
	{
	"amr-nb" : {
	"enabled" : "yes"
	}
		{
		"audio_codecs" : [
		{
		"amr-wb" : {
		"enabled" : "yes"
		}
		},
		{
		"g723" : {
		"enabled" : "yes"
		}
		},
		{
		"amr-nb" : {
		"enabled" : "yes"
		}

Method	Request Payload	Response Payload
	`},{"g729" : {"enabled" : "yes"}},{"pcmu" : {"enabled" : "yes"}},{"pcma" : {"enabled" : "yes"}},{"g722" : {"enabled" : "yes"}},{"g726" : {"enabled" : "yes"}}],"video_codecs" : [{"h264" : {"enabled" : "yes","bitrate" : "768000","fps" : "25","level" : "22","profile" : "66","sample_rate" : "0","size" : "vga"}},{"h263" : {"enabled" : "yes","bitrate" : "384000","fps" : "30","level" : "30","profile" : "0","sample_rate" : "0","size" : "cif"}}]}`	`},{"g729" : {"enabled" : "yes"}},{"pcmu" : {"enabled" : "yes"}},{"pcma" : {"enabled" : "yes"}},{"g722" : {"enabled" : "yes"}},{"g726" : {"enabled" : "yes"}}],"video_codecs" : [{"h264" : {"enabled" : "yes","bitrate" : "768000","fps" : "25","level" : "22","profile" : "66","sample_rate" : "0","size" : "vga"}},{"h263" : {"enabled" : "yes","bitrate" : "384000","fps" : "30","level" : "30","profile" : "0","sample_rate" : "0","size" : "cif"}}]}`

/routing

Routing rules map inbound SIP URIs to their applications. Each rule consists of a pattern and an application name.

Resource URI

```
/routing
```

HTTP GET

Sets and retrieves the routing configuration.

```
GET /routing
```

Method	Response Payload

JSON	<pre>{
	  "routes" : [
	    {
	      "pattern" : "^sip:100.*",
	      "application" : "app"
	    },
	    {
	      "pattern" : "^sip:101.*",
	      "application" : "app"
	    }
	  ]
	}</pre>

HTTP PUT

Modifies the routing configuration.

```
PUT /routing
```

Method	Request Payload	Response Payload

JSON	<pre>{
	  "routes" : [
	    {
	      "pattern" : "^sip:100.*",
	      "application" : "app"
	    },
	    {
	      "pattern" : "^sip:101.*",
	      "application" : "app"
	    }
	    {
	      "pattern" : "^sip:102.*",
	      "application" : "app2"
	    }
	  ]
	}</pre>
		<pre>{
		  "routes" : [
		    {
		      "pattern" : "^sip:100.*",
		      "application" : "app"
		    },
		    {
		      "pattern" : "^sip:101.*",
		      "application" : "app"
		    }
		    {
		      "pattern" : "^sip:102.*",
		      "application" : "app2"
		    }
		  ]
		}</pre>

/msml

Resource URI

```
/msml
```

HTTP GET

Sets and retrieves all MSML server settings.

```
GET /msml
```

Method	Response Payload

JSON	{
	"version": "1.1",
	"http_caching": "no",
	"schema_validation": "no",
	"adaptor port": "32868",
	"storage_directory": "/var/lib/xms/media/en_US",
	"content_type": "xml",
	"encoding": "utf_8",
	"clear_db": "yes",
	"dtmf_start_time": "no",
	"adv_digit_pattern": "no",
	"video_fast_update": "INFO",
	"video bandwidth": "512",
	"conf_agc_default": "no",
	"default_amr_alignment": "BANDWIDTH_EFFICIENT",
	"dtmf_detect_mode": "RFC2833",
	"dns_cache_timeout": "60",
	"cert verify peer": "no",
	"cert_verify_host": "no",
	"cpa": [
	{
	"config1": {
	"cnosig": 40000,
	"no_answer": 30000,
	"pamd_failtime": 4000
	}
	}
	]
	}

HTTP PUT

Modifies the MSML server configuration.

```
PUT /msml
```

Valid "version" settings:

- 1.0
- 1.1

Valid "content_type" settings:

- xml
- msml_xml

Valid "encoding" settings:

- utf_8
- us_ascii

Valid "video_fast_update" settings:

- INFO
- DISABLE

Valid "http_caching", "schema_validation", "clear_db", "dtmf_start_time", "adv_digit_pattern", "cert_verify_peer", and "cert_verify_host" settings:

- yes
- no

Valid "bandwidth_modifier" settings (in kbps):

- None, 48, 64, 128, 256, 400, 512, 800, 1024, 2048, 4096

Valid "dtmf_detect_mode" settings:

- RFC2833
- IN-BAND

Valid "default_amr_alignment" settings:

- BANDWIDTH_EFFICIENT
- OCTET_ALIGNED

Valid "dns_cache_timeout" settings (in seconds):

- 0 means no cache
- -1 means cache forever

CPA (cpa) configuration defines how CPA operates. For instance, the time required before "no answer" is declared. In a MSML script, the "cfgname" attribute is set on the <cpa> element to specify the "config1" configuration as follows.

```
<cpa cfgname="config1"/>
```

Continuous No Signal (cnosig) is the maximum time of silence, with no signal, allowed immediately after cadence detection begins. If exceeded, a "no ringback" MSML event is generated by the media server.

No Answer (no_answer) is the length of time to wait after first ringback before deciding that the call is not answered.

PAMD Fail Time (pamd_failtime) is the maximum time to wait for positive answering machine detection or positive voice detection after a cadence break.

Method	Request Payload	Response Payload

JSON	{
	"http_caching" : "yes",
	"cpa": [
	{
	"config2": {
	"cnosig": 40000,
	"no_answer": 30000,
	"pamd_failtime": 4000
	}
	}
	]
	}
		{
		"version": "1.1",
		"http_caching": "yes",
		"schema_validation": "no",
		"adaptor_port": "32868",
		"storage directory":
		"/var/lib/xms/media/en_US",
		"content_type": "xml",
		"encoding": "utf_8",
		"clear_db": "yes",
		"dtmf_start_time": "no",
		"adv_digit_pattern": "no",
		"video_fast_update": "INFO",
		"video bandwidth": "512",
		"conf_agc_default": "no",
		"default_amr_alignment":
		"BANDWIDTH_EFFICIENT",
		"dtmf detect mode": "RFC2833",
		"dns_cache_timeout": "60",
		"cert_verify_peer": "no",
		"cert_verify_host": "no",
		"cpa": [
		{
		"config2": {
		"cnosig": 40000,
		"no_answer": 30000,
		"pamd_failtime": 4000
		}
		}
		]
		}

/media

Resource URI

```
/media
```

HTTP GET

Retrieves a list of media files and directories.

```
GET /media
```

Method	Response Payload
JSON	{ "locale": "en-US", "name": "", "type": "dir", "uri": "/media", "children": [{ "name": "verification", "type": "dir", "uri": "/media/verification", "children": [{ "name": "play_menu.jpeg", "type": "file", "uri": "/media/verification/play_menu.jpeg" }, { "name": "greeting.jpeg", "type": "file",

Method	Response Payload

	"uri": "/media/verification/greeting.jpeg"
	},
	{
	"name": "record_intro.wav",
	"type": "file",
	"uri": "/media/verification/record_intro.wav"
	},
	{
	"name": "main_menu.jpeg",
	"type": "file",
	"uri": "/media/verification/main_menu.jpeg"
	},
	{
	"name": "greeting.wav",
	"type": "file",
	"uri": "/media/verification/greeting.wav"
	},
	{
	"name": "record_intro.jpeg",
	"type": "file",
	"uri": "/media/verification/record_intro.jpeg"
	},
	{
	"name": "play_menu.wav",
	"type": "file",
	"uri": "/media/verification/play_menu.wav"
	},
	{
	"name": "verification_intro.wav",
	"type": "file",
	"uri": "/media/verification/verification_intro.wav"
	},
	{
	"name": "main menu.wav",
	"type": "file",
	"uri": "/media/verification/main_menu.wav"
	}
	]
	},
	{
	"name": "vxml",
	"type": "dir",
	"uri": "/media/vxml",
	"children": [
	{
	"name": "recorded",
	"type": "dir",
	"uri": "/media/vxml/recorded",
	"children": []
	},
	{
	"name": "generic",
	"type": "dir",
	"uri": "/media/vxml/generic",
	"children": [
	{
	"name": "audio",
	"type": "dir",
	"uri": "/media/vxml/generic/audio",
	"children": [
	{
	"name": "beep.wav",
	"type": "file",
	"uri": "/media/vxml/generic/audio/beep.wav"
	},
	{
	"name": "maxspeechtimeout.wav",

Method	Response Payload


```
                  "type": "file",
                  "uri": "/media/vxml/generic/audio/maxspeechtimeout.wav"
                },
                {
                  "name": "calltimeout.wav",
                  "type": "file",
                  "uri": "/media/vxml/generic/audio/calltimeout.wav"
                },
                {
                  "name": "genericerror.wav",
                  "type": "file",
                  "uri": "/media/vxml/generic/audio/genericerror.wav"
                },
                {\
                  "name": "nomatch.wav",
                  "type": "file",
                  "uri": "/media/vxml/generic/audio/nomatch.wav"
                },
                {
                  "name": "semantic.wav",
                  "type": "file",
                  "uri": "/media/vxml/generic/audio/semantic.wav"
                },
                {
                  "name": "transferaudio.wav",
                  "type": "file",
                  "uri": "/media/vxml/generic/audio/transferaudio.wav"
                },
                {
                  "name": "help.wav",
                  "type": "file",
                  "uri": "/media/vxml/generic/audio/help.wav"
                },
                {
                  "name": "badfetch.wav",
                  "type": "file",
                  "uri": "/media/vxml/generic/audio/badfetch.wav"
                },
                {
                  "name": "exit.wav",
                  "type": "file",
                  "uri": "/media/vxml/generic/audio/exit.wav"
                }
              ]
            }
          ]
        }
      ]
    },
    {
      "name": "black.jpeg",
      "type": "file",
      "uri": "/media/black.jpeg"
    }
  ]
}
```


HTTP POST

Creates and replaces a media file.

Content-Type: "audio/wav" or "video/mp4" or "image/jpeg"

```
POST /media/<path/to/file>
```

HTTP DELETE

Deletes a media file.

```
DELETE /media/<path/to/file>
```

/tones

The tones API is used to configure custom tone detection templates.

A tone template may define either a single tone or dual tones, which may be either continuous or cadenced. Dual tones with frequency components closer than approximately 63Hz cannot be detected, for these cases a single tone definition should be used. A maximum of 20 tone templates may be defined.

Parameters

- **freq1**, frequency 1 in Hz (300Hz to 3.5kHz).
- **fq1dev**, frequency 1 deviation in Hz.
- **freq2**, frequency 2 in Hz (300Hz to 3.5kHz). Set to 0 to define a single tone.
- **fq2dev**, frequency 2 deviation in Hz. Dual tone only.
- **ontime**, tone-on time in milliseconds (minimum 40ms). Set to 0 to define a continuous tone.
- **ontdev**, tone-on time deviation in milliseconds. Cadenced only.
- **offtime**, tone-off time in milliseconds (minimum 40ms). Cadenced only.
- **offtdev**, tone-off time deviation in milliseconds. Cadenced only.

Resource URI

```
/tones
```

HTTP GET

Retrieves all tone templates.

```
GET /tones
```

Method	Response Payload

JSON	<pre>{
	  "tones" : [
	    {
	      "1kHz" : {
	          "freq1" : 1000,
	          "fq1dev" : 20,
	          "freq2" : 0,
	          "fq2dev" : 0,
	          "ontime" : 0,
	          "ontdev" : 0,
	          "offtime" : 0,
	          "offtdev" : 0
	      }
	    },
	    {
	      "busy" : {
	           "freq1" : 480,
	           "fq1dev" : 40,
	           "freq2" : 620,
	           "fq2dev" : 40,
	           "ontime" : 500,
	           "ontdev" : 50,
	           "offtime" : 500,</pre>

Method	Response Payload


```
          "offtdev" : 50
    }
   }
 ],
 "cpa tones": [
   {
     "busy1": {
       "freq1": 480,
       "fq1dev": 30,
       "freq2": 620,
       "fq2dev": 30,
       "twinfreq": 0,
       "twindev": 0,
       "ontime": 500,
       "ontdev": 150,
       "offtime": 500,
       "offtdev": 150,
       "repcnt": 2
     }
   },
   {
     "busy2": {
       "freq1": 480,
       "fq1dev": 30,
       "freq2": 620,
       "fq2dev": 30,
       "twinfreq": 0,
       "twindev": 0,
       "ontime": 250,
       "ontdev": 150,
       "offtime": 250,
       "offtdev": 150,
       "repcnt": 2
     }
   },
   {
     "dialtone_international": {
       "freq1": 340,
       "fq1dev": 40,
       "freq2": 440,
       "fq2dev": 40,
       "twinfreq": 390,
       "twindev": 90,
       "ontime": 1000,
       "ontdev": 0,
       "offtime": 0,
       "offtdev": 0,
       "repcnt": 1
     }
   },
   {
     "dialtone_local": {
       "freq1": 340,
       "fq1dev": 40,
       "freq2": 440,
       "fq2dev": 40,
       "twinfreq": 390,
       "twindev": 90,
       "ontime": 100,
       "ontdev": 0,
       "offtime": 0,
       "offtdev": 0,
       "repcnt": 1
     }
   },
   {
     "fax1": {
```

Method	Response Payload

	"freq1": 1100,
	"fq1dev": 50,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 350,
	"ontdev": 250,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 1
	}
	},
	{
	"fax2": {
	"freq1": 2150,
	"fq1dev": 150,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 100,
	"ontdev": 0,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 1
	}
	},
	{
	"ringback1": {
	"freq1": 440,
	"fq1dev": 50,
	"freq2": 480,
	"fq2dev": 50,
	"twinfreq": 450,
	"twindev": 100,
	"ontime": 2000,
	"ontdev": 200,
	"offtime": 4000,
	"offtdev": 200,
	"repcnt": 1
	}
	},
	{
	"ringback2:seg1": {
	"freq1": 450,
	"fq1dev": 100,
	"freq2": 450,
	"fq2dev": 100,
	"twinfreq": 450,
	"twindev": 100,
	"ontime": 600,
	"ontdev": 400,
	"offtime": 600,
	"offtdev": 400,
	"repcnt": 0
	}
	},
	{
	"ringback2:seg2": {
	"freq1": 450,
	"fq1dev": 100,
	"freq2": 450,
	"fq2dev": 100,
	"twinfreq": 450,
	"twindev": 100,
	"ontime": 600,

Method	Response Payload

	"ontdev": 400,
	"offtime": 3500,
	"offtdev": 2500,
	"repcnt": 1
	}
	},
	{
	"sit_no_circuit:seg1": {
	"freq1": 985,
	"fq1dev": 35,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 385,
	"ontdev": 65,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 0
	}
	},
	{
	"sit_no_circuit:seg2": {
	"freq1": 1425,
	"fq1dev": 25,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 385,
	"ontdev": 65,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 0
	}
	},
	{
	"sit_no_circuit:seg3": {
	"freq1": 1795,
	"fq1dev": 55,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 0,
	"ontdev": 0,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 1
	}
	},
	{
	"sit_operator_intercept:seg1": {
	"freq1": 915,
	"fq1dev": 40,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 225,
	"ontdev": 75,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 0
	}
	},
	{

Method	Response Payload

	"sit operator intercept:seg2": {
	"freq1": 1370,
	"fq1dev": 60,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 225,
	"ontdev": 75,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 0
	}
	},
	{
	"sit_operator_intercept:seg3": {
	"freq1": 1795,
	"fq1dev": 55,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 0,
	"ontdev": 0,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 1
	}
	},
	{
	"sit_reorder:seg1": {
	"freq1": 915,
	"fq1dev": 40,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 225,
	"ontdev": 75,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 0
	}
	},
	{
	"sit_reorder:seg2": {
	"freq1": 1425,
	"fq1dev": 25,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 385,
	"ontdev": 65,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 0
	}
	},
	{
	"sit_reorder:seg3": {
	"freq1": 1795,
	"fq1dev": 55,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,

Method	Response Payload

	"ontime": 0,
	"ontdev": 0,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 1
	}
	},
	{
	"sit_vacant_circuit:seg1": {
	"freq1": 985,
	"fq1dev": 35,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 385,
	"ontdev": 65,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 0
	}
	},
	{
	"sit_vacant_circuit:seg2": {
	"freq1": 1370,
	"fq1dev": 60,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 225,
	"ontdev": 75,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 0
	}
	},
	{
	"sit_vacant_circuit:seg3": {
	"freq1": 1795,
	"fq1dev": 55,
	"freq2": 0,
	"fq2dev": 0,
	"twinfreq": 0,
	"twindev": 0,
	"ontime": 0,
	"ontdev": 0,
	"offtime": 0,
	"offtdev": 0,
	"repcnt": 1
	}
	}
	]
	}

HTTP PUT

Updates all tone templates.

```
PUT /tones
```

Method	Request Payload	Response Payload

JSON	{
	"tones" : [
	{
	"1kHz" : {
	"freq1" : 1000,
	"fq1dev" : 20,
	"freq2" : 0,
	"fq2dev" : 0,
	"ontime" : 0,
	"ontdev" : 0,
	"offtime" : 0,
	"offtdev" : 0
	}
	},
	{
	"busy" : {
	"freq1" : 480,
	"fq1dev" : 40,
	"freq2" : 620,
	"fq2dev" : 40,
	"ontime" : 500,
	"ontdev" : 50,
	"offtime" : 500,
	"offtdev" : 50
	}
	},
	{
	"congestion" : {
	"freq1" : 480,
	"fq1dev" : 40,
	"freq2" : 620,
	"fq2dev" : 40,
	"ontime" : 200,
	"ontdev" : 50,
	"offtime" : 300,
	"offtdev" : 50
	}
	}
	]
	}
		{
		"tones" : [
		{
		"1kHz" : {
		"freq1" : 1000,
		"fq1dev" : 20,
		"freq2" : 0,
		"fq2dev" : 0,
		"ontime" : 0,
		"ontdev" : 0,
		"offtime" : 0,
		"offtdev" : 0
		}
		},
		{
		"busy" : {
		"freq1" : 480,
		"fq1dev" : 40,
		"freq2" : 620,
		"fq2dev" : 40,
		"ontime" : 500,
		"ontdev" : 50,
		"offtime" : 500,
		"offtdev" : 50
		}
		},
		{
		"congestion" : {
		"freq1" : 480,
		"fq1dev" : 40,
		"freq2" : 620,
		"fq2dev" : 40,
		"ontime" : 200,
		"ontdev" : 50,
		"offtime" : 300,
		"offtdev" : 50
		}
		}
		]
		}

/network

Resource URI

```
/network
```

Resources

Resource	Description
interface	Network interface configuration resource
dns	Network DNS configuration resource

HTTP GET

Retrieves all available system resources.

```
GET /network
```

Method	Response Payload

JSON	{
	"resources" : [
	{
	"uri" : "\/network\/interface"
	},
	{
	"uri" : "\/network\/dns"
	}
	]
	}

/network/interface

Resource URI

```
/network/interface
```

HTTP GET

Retrieves all network interfaces on the system.

```
GET /network/interface
```

Method	Response Payload

JSON	{
	"interfaces" : [
	{
	"ifname" : "lo",
	"uri" : "\/network\/interface\/lo"
	},
	{
	"ifname" : "eth0",
	"uri" : "\/network\/interface\/eth0"
	},
	{
	"ifname" : "eth0:0",
	"uri" : "\/network\/interface\/eth0:0"
	}
	],
	}

Single Instance

Retrieves information for a single network interface.

```
GET /network/interface/eth0
```

Method	Response Payload

JSON	<pre>{
	    "if_name" : "192.168.195.12",
	    "ipv4_addr" : "192.168.195.12",
	    "ipv4_mask" : "255.255.255.0",
	    "mac" : "00:21:70:cf:32:b1",
	    "mode" : "dhcp",
	    "dns1" : "10.20.22.1",
	    "dns2" : ""
	}</pre>

HTTP PUT

Configures a single network interface.

```
PUT /network/interface/eth0
```

Method	Request Payload

JSON	<pre>{
	    "if_name" : "eth0"
	    "ipv4_addr" : "10.20.129.12",
	    "ipv4_mask" : "255.255.255.0",
	    "gateway" : "10.20.129.250"
	    "mode" : "static"
	    "dns1" : "10.20.129.1"
	}</pre>

/network/dns

Resource URI

```
/network/dns
```

HTTP GET

Retrieves the DNS information on the system.

```
GET /network/dns
```

Method	Response Payload

JSON	<pre>{
	  "hostname": "xms.localdomain",
	  "search": "dialogic.com",
	  "nameserver": [
	    {
	      "addr": "10.20.106.1"
	    },
	    {
	      "addr": "10.20.106.2"
	    }
	  ]
	}</pre>

HTTP PUT

Configures the DNS information on the system.

```
PUT /network/dns
```

Method	Request Payload

JSON	<pre>{
	   "hostname" : "ab.xyz.com",
	   "search" : "xyz.com",
	   "nameserver":[{
	      "addr": "192.168.23.1"
	    },
	    {
	      "addr": "192.168.23.2"
	    }]
	}</pre>

/mrcpclient

Resource URI

```
/mrcpclient
```

Resources

Resource	Description
global	MRCP Client Global configuration resource
speechserver1	MRCP Client Speechserver1 configuration resource
speechserver2	MRCP Client Speechserver2 configuration resource

HTTP GET

Retrieves all available MRCP Client resources.

```
GET /mrcpclient
```

Method	Response Payload

JSON	<pre>{
	  "resources": [
	    {
	      "uri": "/mrcpclient/global"
	    },
	    {
	      "uri": "/mrcpclient/speechserver1"
	    },
	    {
	      "uri": "/mrcpclient/speechserver2"
	    }
	  ]
	}</pre>

/mrcpclient/global

Resource URI

```
/mrcpclient/global
```

HTTP GET

Retrieves the MRCP Client Global configuration.

```
GET /mrcpclient/global
```

Method	Response Payload

JSON	<pre>{
	  "clientaddress": "0.0.0.0",
	  "keepaliveinterval": "10000",
	  "keepalivecount": "3",
	  "socketconnectionbackoff": "3000",
	  "maxsessions": "100"
	}</pre>

HTTP PUT

Configures the MRCP Client Global resource.

```
PUT /mrcpclient/global
```

Method	Request Payload	Response Payload

JSON	<pre>{
	  "clientaddress": "10.20.110.80",
	  "keepaliveinterval": "20000",
	  "keepalivecount": "4",
	  "socketconnectionbackoff":
	"5000",
	  "maxsessions": "20"
	}</pre>
		<pre>{
		  "clientaddress": "10.20.110.80",
		  "keepaliveinterval": "20000",
		  "keepalivecount": "4",
		  "socketconnectionbackoff":
		"5000",
		  "maxsessions": "20"
		}</pre>

/mrcpclient/speechserver1

Resource URI

```
/mrcpclient/speechserver1
```

HTTP GET

Retrieves the MRCP Client Speechserver1 configuration.

```
GET /mrcpclient/speechserver1
```

Method	Response Payload

JSON	<pre>{
	  "id": "SERVER_1",
	  "protocol": "MRCP/2.0",
	  "session": "SIP/2.0",
	  "transport": "TCP",
	  "address": "127.0.0.1",
	  "port": "5060",
	  "asr": "true",
	  "tts": "false"
	}</pre>

HTTP PUT

Configures the MRCP Client Speechserver1 resource.

```
PUT /mrcpclient/speechserver1
```

Method	Request Payload	Response Payload

JSON	<pre>{
	  "transport": "UDP",
	  "address": "127.0.0.1",
	  "port": "5080",
	  "asr": "false",
	  "tts": "true"
	}</pre>
		<pre>{
		  "id": "SERVER_1",
		  "protocol": "MRCP/2.0",
		  "session": "SIP/2.0",
		  "transport": "UDP",
		  "address": "127.0.0.1",
		  "port": "5080",
		  "asr": "false",
		  "tts": "true"
		}</pre>

/mrcpclient/speechserver2

Resource URI

```
/mrcpclient/speechserver2
```

HTTP GET

Retrieves the MRCP Client Speechserver2 configuration.

```
GET /mrcpclient/speechserver2
```

Method	Response Payload

JSON	<pre>{
	  "id": "SERVER_2",
	  "protocol": "MRCP/2.0",
	  "session": "SIP/2.0",
	  "transport": "TCP",
	  "address": "127.0.0.1",
	  "port": "5060",
	  "asr": "true",
	  "tts": "false"
	}</pre>

HTTP PUT

Configures the MRCP Client speechserver2 resource.

```
PUT /mrcpclient/speechserver2
```

Method	Request Payload	Response Payload

JSON	<pre>{
	  "transport": "UDP",
	  "address": "127.0.0.1",
	  "port": "5080",
	  "asr": "false",
	  "tts": "true"
	}</pre>
		<pre>{
		  "id": "SERVER_2",
		  "protocol": "MRCP/2.0",
		  "session": "SIP/2.0",
		  "transport": "UDP",
		  "address": "127.0.0.1",
		  "port": "5080",
		  "asr": "false",
		  "tts": "true"
		}</pre>

/vxml

Resource URI

```
/vxml
```

Resources

Resource	Description
interpreter	VXML Interpreter configuration resource

HTTP GET

Retrieves all available VXML resources.

```
GET /vxml
```

Method	Response Payload

JSON	{
	  "resources": [
	  {
	  "uri": "/vxml/vxmlinterpreter"
	  }
	  ]
	}

/vxml/vxmlinterpreter

Resource URI

```
/vxml/vxmlinterpreter
```

HTTP GET

Retrieves the VXML Interpreter configuration.

```
GET /vxml/vxmlinterpreter
```

Method	Response Payload

JSON	{
	  "AllowCallTransfer": "true",
	  "DefaultCompleteTimeout": "0.25s",
	  "DefaultGrammarLocale": "en-US",
	  "DefaultIncompleteTimeout": "0.75s",
	  "DefaultInitialURI": "%VXMLROOT%/www/vxml/index.vxml",
	  "DefaultInterDigitTimeout": "3s",
	  "DefaultTimeout": "3.4s",
	  "DefaultTTSLang": "en-US",
	  "NumChannels": "5",
	  "SystemLogLevel": "1",
	  "VXMLAppLogsEnabled": "true",
	  "StaticContentDir": "/var/lib/xms/vxml/www",
	  "WebServerLocalIpAddress": "127.0.0.1",
	  "WebServerListenPort": "9002",
	  "WebServerUsername": "",
	  "WebServerPassword": ""
	}

HTTP PUT

Configures the VXML Interpreter resource.

```
PUT /vxml/vxmlinterpreter
```

Method	Request Payload	Response Payload

JSON	<pre>{
	  "AllowCallTransfer": "false",
	  "SystemLogLevel": "5",
	}</pre>
		<pre>{
		  "AllowCallTransfer": "false",
		  "DefaultCompleteTimeout":
		"0.25s",
		  "DefaultGrammarLocale": "en-US",
		  "DefaultIncompleteTimeout":
		"0.75s",
		  "DefaultInitialURI":
		"%VXMLROOT%/www/vxml/index.vxml",
		  "DefaultInterDigitTimeout": "3s",
		  "DefaultTimeout": "3.4s",
		  "DefaultTTSLang": "en-US",
		  "NumChannels": "5",
		  "SystemLogLevel": "5",
		  "VXMLAppLogsEnabled": "true",
		  "StaticContentDir":
		"/var/lib/xms/vxml/www",
		  "WebServerLocalIpAddress":
		"127.0.0.1",
		  "WebServerListenPort": "9002",
		  "WebServerUsername": "",
		  "WebServerPassword": ""
		}</pre>

/xmsrest

Resource URI

```
/xmsrest
```

Resources

Resource	Description
general	RESTful call and media API general configuration resource
trusted	RESTful call and media API trusted application configuration resource

HTTP GET

Retrieves all available XMSREST resources.

```
GET /xmsrest
```

Method	Response Payload

JSON	<pre>{
	  "resources": [
	    {
	      "uri": "/xmsrest/general"
	    },
	    {
	      "uri": "/xmsrest/trusted"
	    }
	  ]
	}</pre>

/xmsrest/general

Resource URI

```
/xmsrest/general
```

HTTP GET

Retrieves the XMSREST General configuration.

```
GET /xmsrest/general
```

Method	Response Payload

JSON	<pre>{
	  "port": "81"
	}</pre>

HTTP PUT

Configures the XMSREST General resource.

```
PUT /xmsrest/general
```

Method	Request Payload	Response Payload

JSON	<pre>{
	  "port": "8181"
	}</pre>
		<pre>{
		  "port": "8181"
		}</pre>

/xmsrest/trusted

Resource URI

```
/xmsrest/trusted
```

HTTP GET

Retrieves all trusted apps.

```
GET /xmsrest/trusted
```

Method	Response Payload

JSON	<pre>{
	  "app": "enable",
	  "app2": "enable"
	}</pre>

HTTP PUT

Adds, deletes, or modifies trusted apps.

```
PUT /xmsrest/trusted
```

Method	Request Payload	Response Payload

JSON	<pre>{
	  "app3": "enable",
	  "app2": "disable"
	}</pre>
		<pre>{
		  "app3": "enable",
		  "app2": "disable"
		}</pre>