Cisco Digital Media Suite 5.2.3

API Programmer Guide for Cisco Show and Share 5.2.3

Version 0.8
Revised: May 23, 2011

Contents

About This Guide

Cisco Show and Share is one component of *Cisco Digital Media Suite* (Cisco DMS).

This guide describes Cisco APIs for third-party application developers whose products should extend or interoperate with Cisco Show and Share. These APIs are a mechanism to insert, retrieve, update, and remove data.

Similar guides on Cisco.com describe the APIs for other components of Cisco Digital Media Suite, such as Cisco Digital Signs.

Who Should Use This Guide

This guide is a technical resource for application developers who build custom user interfaces, workflows, and vertical-specific applications that extend Cisco Show and Share.

You should have an advanced level of understanding of web technology, operation, and terminology and be familiar with Cisco Show and Share.

Application developers who use this application programming interface (API) should also have an understanding of a high-level programming language, such as Java or an equivalent language. Additionally, we recommend that you have knowledge of the following:

- XML and XML Schema
- Representational State Transfer (REST) and RESTful Services

You should be familiar with using Cisco Show and Share. In most cases, API operations correlate to GUI operations.

List of Changes from 5.2.2

This section is for developers that are familiar with Show and Share 5.2.2 API documentation and want to know what has changed in 5.2.3. Listed below are the changes made in this release along with a brief description. You will also find the page number on which the change is explained in further detail. If you have not used a previous version of Show and Share API to integrate with your application, you may skip this section.

Added

API call that uploads a video as a draft into Show and Share in one step.

Checks the status of an MXE transcode job

Changed

Because the single step upload will not work with all clients, the 4-step upload process will still be available. However, there are some changes that have been made to this.

Removed

User and user group management will now be done through a separate list of APIs that are to be made to the DMM. These API are called Platform Service API's and can be found in a separate document.

Introduction to Cisco DMS APIs

This API uses a RESTful architecture, which leverages XML, HTTP, and HTTPS.

You can use this API to perform some of the same operations that are available in the web-based user interface for Cisco Show and Share.

The Show and Share APIs send XML data using an HTTPS POST method to the same constant URL for each API call:

https://<ShowAndShare_Host_Name>:443/vportal/services/xml/api

The server returns an XML response, which is also an encoded REST message, to indicate if the request is successful, or to return data.

You can use this API to perform some of the operations that are available in the Show and Share GUI.

NOTE
Our implementations of Cisco DMS APIs may change over time in response to the changing needs of our partner community. We will maintain backward compatibility whenever possible but advise you to **expect differences in future releases**. A list of changes will be provided for each release to keep API users aware of any necessary code changes that they will need to make.

Getting Started

Topics in this section describe how to begin provisioning services with this API, verify the status of the API, and learn the steps in a typical workflow.

Installation Notes

Cisco Show and Share includes all components for its own API. Aside from the requirement for a *Cisco Show and Share* appliance that is configured and operating correctly, this API imposes no additional startup or shutdown requirements.

The only user interfaces for this API are XML-encoded messages.

Before You Begin

ATTENTION
All applications that use API calls for Cisco Show and Share must pass their XML data with the content type set to **application/xml**.

Checking the API Status

You can verify the status of the API

NOTE
Cisco Systems neither supports nor offers any warranties for DMS API Tester. Our partner community built and maintains this software utility.

To test the APIs:

1. Download the Adobe AIR runtime from http://get.adobe.com/air/ and install it on your computer.
2. Download **DMS_API_Tester.air** from http://developer.cisco.com
3. Install **DMS_API_Tester.air**., and then open it.
4. Complete these steps in DMS API Tester:
   a. Enter the fully qualified domain name (FQDN) of your Show and Share appliance in the Hostname field. For example: *showandshare.example.com*
   b. Enter the administrator's username. For example: *superuser*
   c. Enter the administrator's password.
   d. Choose **/vportal/services/xml/** From the Service list.
   e. Choose **category_getAll** from the Template list.
   f. Click **Submit**.
   g. Accept the security alert.
   h. Verify that you receive a response like this one:

```
<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns">
	<vp:vportal>
		<vportal_id>1</vportal_id>
	</vp:vportal>
	<vp:vprequest>
		<query>getCategories</query>
	</vp:vprequest>
	<vp:vpresponse>
		<user>superuser</user>
		<response>true</response>
		<request_xml><![CDATA[<xml-fragment
xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns">
  <vp:vportal>
    <vportal_id>1</vportal_id>
  </vp:vportal>
  <vp:vprequest>
    <query>getCategories</query>
  </vp:vprequest>
</xml-fragment>]]></request_xml>
	</vp:vpresponse>
	<vp:vpcategory>
		<categoryid>DMS$6</categoryid>
		<name>1</name>
		<path>/dms/core/contentcatalog/vportal/1</path>
	</vp:vpcategory>
</xml-fragment>
```

Workflow

Each Show and Share API call requires user through the HTTP Basic Authentication method. Your application should pass in the username and password on ***each*** API call to guarantee a valid logged in user.

All Show and Share APIs (except for the multipart file uploads) support and should use HTTPS over port 443. Port 8443 is also available for backwards compatibility from 5.2, but we urge developers to use port 443 as there is no guarantee as to how long this will be supported.

ATTENTION
Some API calls require a particular sequence to be followed in order to effectively accomplish them; these are detailed below on a case by case basis.

Making a Web Service Call

The simplest way to make a web service call is to install the DMS API Tester AIR application (see **Checking the API Status** above).

This application will allow you test each of the API calls described below.

ATTENTION
By default, Show and Share uses a self-signed SSL certificate. This triggers a security alert when the first request is made using the DMS API Tester. You must accept this warning manually every time you launch the DMS API Tester and you must also ensure programmatically that your API calls accept this self-signed certificate.

Show and Share APIs

This chapter describes the Cisco Digital Media Suite APIs for operations that are specific to Show and Share.

All Show and Share APIs consist of making an HTTP/HTTPS POST with content type set to “application/xml” and with an XML payload as part of the POST body to a constant URL:

https://<Show_and_Share_Host_Name>:443/vportal/services/xml/api

The XML payload will always contain a root level element called xml-fragment with the “vp” namespace “http://model.data.core.vportal.cisco.com/vp_ns” and at least three child elements, “vp:vportal” (always set to “1” in DMS 5.2), “vp:vprequest” and “query”.

EXAMPLE

```
<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns">

        <vp:vportal>
             <vportal_id>1</vportal_id>
        </vp:vportal>

        <vp:vprequest>
             <query>getCategories</query>
        </vp:vprequest>

</xml-fragment>
```

Category Provisioning

Get Categories

This API call returns all Categories in the form of a tree.

QUERY	**getCategories**
PREREQUISITES	N/A
NOTES	N/A
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vprequest> <query>getCategories</query> </vp:vprequest> </xml-fragment>
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

Content Provisioning

Content in Show and Share is the container of a video along with all of the related assets and metadata.

Content contains a title, description and tags of a video. It also contains information about the media files that constitute the content: video file, thumbnails etc… The last thing content contains is taxonomy and authorization information: how the content is organized on Show and Share and who can access it.

Content goes through the following states:

1. DRAFT: when only the author and approvers of the content can view the content.
2. APPROVED: when the content is ready to be published
3. PUBLISHED: when the content has been published

Get Content details

This API is used to get all the details on a specific piece of Show and Share content.

QUERY	**getContentById**
PREREQUISITES	**ccsid:** the content id
NOTES	N/A
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vprequest> <query>getContentById</query> </vp:vprequest> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vpcontent> <id> <ccsid>____CONTENT_ID____</ccsid> </id> </vp:vpcontent> </xml-fragment>
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

Create Video On Demand (VOD) content in Draft State : Single-Step

This API is used to create content in Show and Share using a video file. If there is an MXE that has been configured in Show and Share, a video uploaded using this API call will automatically be transcoded. A video may only be uploaded by a user that has Author privileges.

QUERY	**Video file upload is done through a multipart-form-data POST request to:** https://<SHOW_AND_SHARE_FQDN>/vportal/services/upload/multipart
PREREQUISITES	**TITLE**: The title of the video **DESCRIPTION**: A description of the video **TAGS**: Tags for the video, separated by commas.
NOTES	• The xml payload <partname> tags map the form attachments to where they will be uploaded. • If a transcript or attachment is not intended to be uploaded, remove the <transcript> or <attachment> tag, any content within these tags, and from the corresponding <input> tag in the html form.
TEMPLATE	An example can be accomplished by copying the HTML below to create a web page, fill the form and submit it.

```
================================================
<html>
        <head>
        <title>Upload</title>
        </head>

        <body>
                <form method='POST' enctype='multipart/form-data'
action='https://<ShowAndShareFqdn>/vportal/services/upload/multipart'>
                        XML:
                        <br>
                        <textarea id='xmlinput' rows=50 cols=100
name=note></textarea>
                        <br>
                        Video: <input type=file name=video1><br>
                        Transcript: <input type=file
name=transcript1><br>
                        Attachment: <input type=file
name=attachment1><br>
                        Attachment2: <input type=file
name=attachment2><br>
                        Attachment3: <input type=file
name=attachment3><br>
                        <br>
                        <input type=submit value=Press> to upload the
file!
                </form>
        </body>
</html>
================================================
```

"xmlinput" should be of the XML format:

```
<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns">
        <vp:vprequest>
                <query>uploadContent</query>
        </vp:vprequest>
        <vp:vportal>
                <vportal_id>1</vportal_id>
        </vp:vportal>
        <vp:vpcontent>
                <title>___TITLE___</title>
                <description>___DESCRIPTION___</description>
                <video>
                        <partname>video1</partname>
                        <transcript>
                                <partname>transcript1</partname>
                        </transcript>
                </video>
                <attachment>
                        <partname>attachment1</partname>
                </attachment>
```

	<attachment> partname>attachment2</partname> </attachment> <attachment> <partname>attachment3</partname> </attachment> <contentattribute> <key>tags</key> <value>___TAGS___</value> <namespace>com.cisco.vportal.1</namespace> </contentattribute> </vp:vpcontent> </xml-fragment>
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

Check Content Transcode Status

This API is only necessary to make if an MXE is enabled. If this is the case and the single-step upload API is used, then the video will begin to transcode automatically upon upload. Because the video cannot be published before the transcoding job is complete, the API user must poll the status to check if they may in fact publish the video or not.

QUERY	getTransformStatus
PREREQUISITES	**CCSID**: The cssid of a video is a unique internal identification number for each video.
NOTES	To reduce resource drain, it is advisable to poll over more spread out intervals. It is recommended that the shortest
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vprequest> <query>getTransformStatus</query> </vp:vprequest> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vpTransformStatusRequest> <contentId> <ccsid>___CCSID___</ccsid> </contentId> </vp:vpTransformStatusRequest> </xml-fragment>
RESULT	The <status> tag in the result xml indicates the transcoding job status. Possible job statuses are: **<status>dms.job.status.skipped</status>** There is no MXE or the transcoding job was never triggered. **<status>dms.job.status.processing</status>** There is no MXE or the transcoding job was never triggered. **<status>dms.job.status.succeeded</status>** There is no MXE or the transcoding job was never triggered.

Create Video On Demand (VOD) content in Draft State : Four-Step

This API is used to create content in Show and Share using a video file. Videos uploaded using this API will NOT automatically be transcoded, even if an MXE has been configured in the Show and Share setup. A video may only be uploaded by a user that has Author privileges.

To upload a video:

1. Create Content with only metadata (no media files)
2. Get a file upload access ticket
3. Upload VOD files to get media urls
4. Update content with media urls

QUERY	**addContent**
PREREQUISITES	**title:** the title of the VOD **description:** the description of the VOD **Epoch Time:** should be set to today's epoch time (milliseconds since January 1, 1970) **tags:** optional attribute that sets tags for the VOD. If undesired, remove the <contentattribute> tag that for this along with its contents.
NOTES	**Step 1 of 4: create content with just metadata.**
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vprequest> <query>addContent</query> </vp:vprequest> <vp:vpcontent> <title>____TITLE____</title> <description>____DESCRIPTION____</description> <contentattribute> <key>addedDate</key> <value>____EPOCH_TIME____</value> <namespace>com.cisco.vportal.1</namespace> </contentattribute> <contentattribute> <key>tags</key> <value>____TAGS____</value> <namespace>com.cisco.vportal.1</namespace> </contentattribute> <contentattribute> <key>allowEmbed</key> <value>true</value> <namespace>com.cisco.vportal.1</namespace> </contentattribute> <contentattribute> <key>allowTags</key> <value>true</value> <namespace>com.cisco.vportal.1</namespace> </contentattribute> <contentattribute> <key>allowRatings</key> <value>true</value> <namespace>com.cisco.vportal.1</namespace> </contentattribute> <contentattribute> <key>allowComments</key> <value>true</value> <namespace>com.cisco.vportal.1</namespace> </contentattribute> <contentattribute> <key>rating</key> <value>0</value>

	<namespace>com.cisco.vportal.1</namespace> </contentattribute> <contentattribute> <key>ratingCount</key> <value>0</value> <namespace>com.cisco.vportal.1</namespace> </contentattribute> <contentattribute> <key>discussedCount</key> <value>0</value> <namespace>com.cisco.vportal.1</namespace> </contentattribute> <contentattribute> <key>viewCount</key> <value>0</value> <namespace>com.cisco.vportal.1</namespace> </contentattribute> <live>false</live> <state> <contentstate>DRAFT</contentstate> </state> </vp:vpcontent> </xml-fragment>
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

QUERY	**getAccessTicket**
PREREQUISITES	N/A
NOTES	**Step 2 of 4: get a file upload ticket**
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vprequest> <query>getAccessTickets</query> </vp:vprequest> <vpTicketRequest> <ticketUri>GenericFileUpload</ticketUri> <ticketQuantity>1</ticketQuantity> </vpTicketRequest> </xml-fragment>
RESULT	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vpresponse> <user>superuser</user> <response>true</response> </vp:vpresponse> <vpTicket> <value>dms_dummy_token</value> </vpTicket> </xml-fragment>

QUERY	**Video file upload is done through a multipart-form-data POST request to:** http://<HOST_NAME>:8080/vportal/GenericFileUpload
PREREQUISITES	**portalId**: should always be equal to "1" for this release **contentId**: should be the contented from step 1 **responseMode**: set to "xml" to get an xml response (otherwise will default to json) **ticket**: in the format of "ticket:username" where ticket is the value from step2 and username is the same username being used to make the API calls **unpackZips**: In case you upload a zip file, it is set to "false" to keep the zip file intact on the server or "true" if you wish the zip file to be unpacked (will return a list of file tracker xmls, one for each file) **ATTENTION:** This API only accepts a single uploaded file in the post. The file does not have to be using any particular key parameter but the post must be constructed in such a way that the contentId is encountered in the data stream before the file data
NOTES	**Step 3 of 4: upload files**
TEMPLATE	An example can be accomplished by copying the HTML below to create a web page, fill the form and submit it. ========== `<html><body>` `<form METHOD=POST` `action="https://SNS_FQDN:443/vportal/GenericFileUploadProtected"` `ENCTYPE="multipart/form-data">` `<p><input type="hidden" name=portalId value="1">` `<p><div style="width:500;text-align:center;font-size:18px">Show-N-Share Upload API tester</div>` `<p>ticket: <input type="text" name="ticket"` `value="dms_dummy_token:username"> ` `<p>contentId: <input type="text" name="contentId"` `value="____CONTENT_ID____"> ` `<p><input type="hidden" name="unpackZips" value="false">` `<p><input type="hidden" name="responseMode" value="xml">` `<p>Choose a File: <input type="file" name="FileData"` `value=""> ` `<input type=submit value="Upload File">` `</form>` `<p>File will be uploaded to: ` `<b><script>document.write(document.forms[0].action);</script></b>` ` (edit the "action" parameter in this file to change the upload url)` `</body></html>` ==========
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

QUERY	**updateContent**
PREREQUISITES	Infomration from step 3 videoformat: is MPEG for .flv and h.264 files, it is WMV for windows media files. **NOTE:** In the template below, the string after ":" in the variable place holders is the XPath of the variables from step 3.
NOTES	**Step 4 of 4: update content with information from Step 3**
TEMPLATE	<xml-fragment> <vp:vprequest xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <query>updateContent</query> </vp:vprequest> <vp:vportal xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vportal_id>1</vportal_id> </vp:vportal> <vp:vpcontent xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <contentattribute> <key>duration</key> <value>____DURATION:UpdatedObject/duration___</value> <namespace>com.cisco.vportal.1</namespace> </contentattribute> <contentattribute> <key>videoformat</key> <value>____MPEG_or_WMV____</value> <namespace>com.cisco.vportal.1</namespace> </contentattribute> <id> <ccsid>____CONTENT_ID____</ccsid> </id> <title>____CONTENT_TITLE____</title> <description>____CONTENT_DESCRIPTION____</description> <live>false</live> <state> <contentstate>DRAFT</contentstate> </state> <video> <live>false</live> <videoasset> <external>false</external> <relativefilepath>____RELATIVE_FILE_PATH:UpdatedObject/FileName____</relativefilepath> <filesize>____FILE_SIZE:UpdatedObject/FileSize____</filesize> <format> <videoformat>____MPEG_or_WMV____</videoformat> </format> <bitrate>____BITRATE:UpdatedObject/bitrate____</bitrate> <framerate>____FRAMERATE:UpdatedObject/framerate____</framerate> <height>____HEIGHT:UpdatedObject/height___</height> <width>____WIDTH:UpdatedObject/width___</width> </videoasset> <image> ====COPY ENTIRE CONTENT OF XML NODE:UpdatedObject/image====

```
                        </image>

        <duration>____DURATION:UpdatedObject/duration
____</duration>
                </video>
                <contentthumbnail>
====COPY_ENTIRE_CONTENT_OF_XML_NODE:UpdatedObject/image====
====EXCEPT_CHANGE_CONTENT_OF_"USAGE"_TAG_TO:
CONTENT_THUMBNAIL=====
                </contentthumbnail>
                <contentattribute>
                        <key>ratingCount</key>
                        <value>0</value>
                        <namespace>com.cisco.vportal.1</namespace>
                </contentattribute>
                <contentattribute>
                        <key>discussedCount</key>
                        <value>0</value>
                        <namespace>com.cisco.vportal.1</namespace>
                </contentattribute>
                <contentattribute>
                        <key>allowRatings</key>
                        <value>true</value>
                        <namespace>com.cisco.vportal.1</namespace>
                </contentattribute>
                <contentattribute>
                        <key>addedDate</key>
                        <value>____EPOCH_TIME____</value>
                        <namespace>com.cisco.vportal.1</namespace>
                </contentattribute>
                <contentattribute>
                        <key>viewCount</key>
                        <value>0</value>
                        <namespace>com.cisco.vportal.1</namespace>
                </contentattribute>
                <contentattribute>
                        <key>allowTags</key>
                        <value>true</value>
                        <namespace>com.cisco.vportal.1</namespace>
                </contentattribute>
                <contentattribute>
                        <key>tags</key>
                        <value>____TAGS____</value>
                        <namespace>com.cisco.vportal.1</namespace>
                </contentattribute>
                <contentattribute>
                        <key>rating</key>
                        <value>0</value>
                        <namespace>com.cisco.vportal.1</namespace>
                </contentattribute>
                <contentattribute>
                        <key>allowComments</key>
                        <value>true</value>
                        <namespace>com.cisco.vportal.1</namespace>
                </contentattribute>
                <contentattribute>
                        <key>allowEmbed</key>
                        <value>true</value>
                        <namespace>com.cisco.vportal.1</namespace>
                </contentattribute>
        </vp:vpcontent>
</xml-fragment>
```

RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

Publish Draft Content to a particular Category

Content is only accessible once it has been published to a particular category.

This is done in four steps:

1. Change content state to “APPROVED”
2. Set content categories
3. Set content access permissions
4. Publish content

QUERY	**changeContentState**
PREREQUISITES	**ccsid**: the content id
NOTES	**Step 1 of 4: Change content state to “APPROVED”**
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vprequest> <query>changeContentState</query> </vp:vprequest> <vp:vpcontent> <id> <ccsid>____CONTENT_ID_____</ccsid> </id> <state> <contentstate>APPROVED</contentstate> </state> </vp:vpcontent> </xml-fragment>
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

QUERY	**setCategoriesForContent**
PREREQUISITES	**ccsid**: the content id **Category path**: Path of the Category that the content is associated with
NOTES	**Step 2 of 4: Set content Categories**
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vprequest> <query>setCategoriesForContent</query> </vp:vprequest> <vp:vpcontent> <id> <ccsid>____CONTENT_ID____</ccsid> </id> </vp:vpcontent> <vp:vpcategory> <child> <path>____CATEGORY_PATH____</path> </child> </vp:vpcategory> </xml-fragment>
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

QUERY	**saveContentViewership**
PREREQUISITES	**ccsid**: the content id **username**: the username of the users that have access to the content
NOTES	**Step 3 of 4: Set content access permissions** **You can skip this step if you are publishing a public video.**
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vprequest> <query>saveContentViewership</query> </vp:vprequest> <vp:vpcontent> <id> <ccsid>____CONTENT_ID____</ccsid> </id> <viewers> <ispublic>false</ispublic> <usergroup> <user> <username>____USERNAME____</username> </user> </usergroup> </viewers> </vp:vpcontent> </xml-fragment>
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

QUERY	**deploy**
PREREQUISITES	**ccsid**: the content id
NOTES	**Step 4 of 4: Publish content**
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vprequest> <query>deploy</query> </vp:vprequest> <vp:vpcontent> <id> <ccsid>____CONTENT_ID____</ccsid> </id> </vp:vpcontent> <vp:vpdeployRequest> <immediate>true</immediate> </vp:vpdeployRequest> </xml-fragment>
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

Remove Content

This API is used to remove Show and Share content.

QUERY	**removeContent**
PREREQUISITES	**ccsid**: the content id
NOTES	N/A
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vprequest> <query>removeContents</query> </vp:vprequest> <vp:vpcontent> <id> <ccsid>____CONTENT_ID____</ccsid> </id> </vp:vpcontent> </xml-fragment>
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

Search Content

This API is used to search all of the Show and Share content and return an ordered list of content.

The supported searches include both published and draft versions of:

- Finding most recently added content
- Finding most viewed content
- Finding content belonging to a specific author
- Finding content belonging to a specific category
- Finding content matching a string

NOTE
In the examples below, you can change from searching for “PUBLISHED” or “DRAFT” states by changing the contentState field’s value.

QUERY	**searchContent**
PREREQUISITES	**start**: is the results index at which the returned content list should start **limit**: is the results index at which the returned content list should stop (always > "start") **startValue**: should be set to today's epoch time (milliseconds since January 1, 1970)
NOTES	**Find most recent content**
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vprequest> <query>searchContent</query> </vp:vprequest> <vp:vpcontentsearch> <locale>en_US</locale> <start>____RESULTS_OFFSET_____</start> <limit>____MAX_RESULTS____</limit> <contentType>VIDEO_PORTAL</contentType> <searchQuery> <searchText/> <fields> <field>title</field> <field>description</field> <field>author</field> <field>com.cisco.vportal.1.tags</field> </fields> <searchParamList> <searchParam> <fieldName>searchType</fieldName> <fieldValue>content/composite/vp</fieldValue> <paramClause>EQUAL</paramClause> <boost>0.9</boost> </searchParam> <searchParam> <fieldName>contentState</fieldName> <fieldValue>PUBLISHED</fieldValue> <paramClause>EQUAL</paramClause> <boost>0.9</boost> </searchParam> </searchParamList> <sortCriteriaList> <sortCriteria> <fieldName>com.cisco.vportal.1.addedDate</fieldName> <sortingOrder>DESCENDING</sortingOrder> <sortingPriority>0.1</sortingPriority> </sortCriteria> </sortCriteriaList> <queryRangeList> <queryRange> <fieldName>expirationDate</fieldName> <startValue>1262563624186</startValue> <endValue>9999999999999</endValue> <inclusive>true</inclusive> <boost>0.9</boost> </queryRange> </queryRangeList> </searchQuery> </vp:vpcontentsearch> </xml-fragment>
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

QUERY	**searchContent**
PREREQUISITES	**start**: is the results index at which the returned content list should start **limit**: is the results index at which the returned content list should stop (always > "start") **startValue**: should be set to today's epoch time (milliseconds since January 1, 1970)
NOTES	**Find most viewed content**
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vprequest> <query>searchContent</query> </vp:vprequest> <vp:vpcontentsearch> <locale>en_US</locale> <start>____RESULTS_OFFSET_____</start> <limit>____MAX_RESULTS____</limit> <contentType>VIDEO_PORTAL</contentType> <searchQuery> <searchText/> <fields> <field>title</field> <field>description</field> <field>author</field> <field>com.cisco.vportal.1.tags</field> </fields> <searchParamList> <searchParam> <fieldName>searchType</fieldName> <fieldValue>content/composite/vp</fieldValue> <paramClause>EQUAL</paramClause> <boost>0.9</boost> </searchParam> <searchParam> <fieldName>contentState</fieldName> <fieldValue>PUBLISHED</fieldValue> <paramClause>EQUAL</paramClause> <boost>0.9</boost> </searchParam> </searchParamList> <sortCriteriaList> <sortCriteria> <fieldName>com.cisco.vportal.1.viewCount</fieldName> <sortingOrder>DESCENDING</sortingOrder> <sortingPriority>0.1</sortingPriority> </sortCriteria> </sortCriteriaList> <queryRangeList> <queryRange> <fieldName>expirationDate</fieldName> <startValue>1262563510015</startValue> <endValue>9999999999999</endValue> <inclusive>true</inclusive> <boost>0.9</boost> </queryRange> </queryRangeList> </searchQuery> </vp:vpcontentsearch> </xml-fragment>
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

QUERY	**searchContent**
PREREQUISITES	**start**: is the results index at which the returned content list should start **limit**: is the results index at which the returned content list should stop (always > "start") **startValue**: should be set to today's epoch time (milliseconds since January 1, 1970)
NOTES	**Find content belonging to a specific author**
TEMPLATE	<xml-fragment xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns"> <vp:vportal> <vportal_id>1</vportal_id> </vp:vportal> <vp:vprequest> <query>searchContent</query> </vp:vprequest> <vp:vpcontentsearch> <locale>en_US</locale> <start>____RESULTS_OFFSET_____</start> <limit>____MAX_RESULTS____</limit> <contentType>VIDEO_PORTAL</contentType> <searchQuery> <searchText/> <fields> <field>title</field> <field>description</field> <field>author</field> <field>com.cisco.vportal.1.tags</field> </fields> <searchParamList> <searchParam> <fieldName>searchType</fieldName> <fieldValue>content/composite/vp</fieldValue> <paramClause>EQUAL</paramClause> <boost>0.9</boost> </searchParam> <searchParam> <fieldName>author</fieldName> <fieldValue>____AUTHOR_USERNAME____</fieldValue> <paramClause>EQUAL</paramClause> <boost>0.9</boost> </searchParam> </searchParamList> <queryRangeList/> <queryPhraseList/> <sortCriteriaList/> </searchQuery> <contentGroupId/> </vp:vpcontentsearch> </xml-fragment>
RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

QUERY	**searchContent**
PREREQUISITES	**start**: is the results index at which the returned content list should start **limit**: is the results index at which the returned content list should stop (always > "start") **startValue**: should be set to today's epoch time (milliseconds since January 1, 1970)
NOTES	**Find content belonging to a specific category**
TEMPLATE	

```
<xml-fragment
xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns">
      <vp:vportal>
            <vportal_id>1</vportal_id>
      </vp:vportal>
      <vp:vprequest>
            <query>searchContent</query>
      </vp:vprequest>
      <vp:vpcontentsearch>
            <locale>en_US</locale>
            <start>____RESULTS_OFFSET_____</start>
            <limit>____MAX_RESULTS____</limit>
            <contentType>VIDEO_PORTAL</contentType>
            <searchQuery>
                  <searchText/>
                  <fields>
                        <field>title</field>
                        <field>description</field>
                        <field>author</field>
                        <field>com.cisco.vportal.1.tags</field>
                  </fields>
                  <searchParamList>
                        <searchParam>
                              <fieldName>searchType</fieldName>

<fieldValue>content/composite/vp</fieldValue>
                              <paramClause>EQUAL</paramClause>
                              <boost>0.9</boost>
                        </searchParam>
                        <searchParam>

<fieldName>contentState</fieldName>
                              <fieldValue>PUBLISHED</fieldValue>
                              <paramClause>EQUAL</paramClause>
                              <boost>0.9</boost>
                        </searchParam>
                  </searchParamList>
                  <sortCriteriaList/>
                  <queryRangeList>
                        <queryRange>

<fieldName>expirationDate</fieldName>

<startValue>1262564865332</startValue>
                              <endValue>9999999999999</endValue>
                              <inclusive>true</inclusive>
                              <boost>0.9</boost>
                        </queryRange>
                  </queryRangeList>
            </searchQuery>

<contentGroupId>____CATEGORY_PATH____</contentGroupId>
      </vp:vpcontentsearch>
</xml-fragment>
```

RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

QUERY	**searchContent**
PREREQUISITES	**start**: is the results index at which the returned content list should start **limit**: is the results index at which the returned content list should stop (always > "start") **startValue**: should be set to today's epoch time (milliseconds since January 1, 1970)
NOTES	**Search all content for a string**
TEMPLATE	

```
<xml-fragment
xmlns:vp="http://model.data.core.vportal.cisco.com/vp_ns">
      <vp:vportal>
            <vportal_id>1</vportal_id>
      </vp:vportal>
      <vp:vprequest>
            <query>searchContent</query>
      </vp:vprequest>
      <vp:vpcontentsearch>
            <locale>en_US</locale>
            <start>____RESULTS_OFFSET_____</start>
            <limit>____MAX_RESULTS____</limit>
            <contentType>VIDEO_PORTAL</contentType>
            <searchQuery>

<searchText>____STRING_TO_SEARCH____</searchText>
                  <fields>
                        <field>title</field>
                        <field>description</field>
                        <field>author</field>
                        <field>com.cisco.vportal.1.tags</field>
                  </fields>
                  <searchParamList>
                        <searchParam>
                              <fieldName>searchType</fieldName>

<fieldValue>content/composite/vp</fieldValue>
                              <paramClause>EQUAL</paramClause>
                              <boost>0.9</boost>
                        </searchParam>
                  </searchParamList>
                  <queryRangeList/>
                  <queryPhraseList/>
                  <sortCriteriaList/>
            </searchQuery>
            <contentGroupId/>
      </vp:vpcontentsearch>
</xml-fragment>
```

RESULT	The intended audience for this programmer guide can anticipate and understand the result that this API call returns. Nonetheless, we might document this result in a future revision to this guide.

RSS

RSS feeds can be created by doing a GET request at this url:
https://HOST_NAME:8443/vportal/services/xml/rss/getContentForRSS?<params>

Replacing HOST_NAME with the server name and <params> with URL params of the name value pairs below:

1. vportal_id=1 (always 1 in this release)
2. sort_criteria=rating/viewCount/addedDate/discussedCount/featured/duration
3. sort_order=ASCENDING/DESCENDING
4. search_text=<string you want to search>
5. tag=<tags you want the content to contain>
6. category=<categories you want the content to belong to>
7. start_date=<epoch GMT date>
8. end_date=<epoch GMT date>
9. favorites=true ? MyFavorites
10. live=true ?MyLiveEvents
11. live=false ? MyVODs
12. published_state=<PUBLISHED or DRAFT>
13. user_id=<the is of a specific user>

Embedding a video

A video can be embedded on any webpage by including the snippet of code below:

EXAMPLE

```
<script type="text/javascript"
src="https://<YOUR_HOST_NAME>:8443/vportal/scripts/videoplayer/DmsEmbedLib.js">
</script>
<script>
document.write(dmsEmbed.getSimplePlayer("[[CONTENT_ID]]","https://<YOUR_HOST_NAME>
:8443/vportal/VideoPlayer.jsp?ccsid=[[CONTENT_ID]]#"));
</script>
```

<YOUR_HOST_NAME> should be replaced by your Show and Share Fully Qualified Domain name and both instances of [[CONTENT_ID]] should be replaced with the content id of the video you want to embed on your page.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV
Amsterdam, The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

Printed in USA CXX-XXXXXX-XX 12/08