Video

Plug-in User Guide

1.2.1.162046


vizrt

Disclaimer

Vizrt provides this publication “as is” without warranty of any kind, either expressed or implied.

This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document to ensure that it contains accurate and up-to-date information, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained in this document.

Vizrt’s policy is one of continual development, so the content of this document is periodically subject to be modified without notice. These changes will be incorporated in new editions of the publication. Vizrt may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

Vizrt may have patents or pending patent applications covering subject matters in this document. The furnishing of this document does not give you any license to these patents.

Technical Support

For technical support and the latest news of upgrades, documentation, and related products, visit the Vizrt web site at www.vizrt.com.

Last Updated

10.12.2014


vizrt

Table of Contents


vizrt

1 Introduction

The Escenic Content Engine's Video plug-in supports integration of the Content Engine with **Viz Media Engine (VME)**, making it possible to:

- Access video content archived in a Viz Media Engine from Escenic web sites
- Use Viz Media Engine to:
  - transcode video content
  - generate **keyframes** from video content
  - send transcoded video content to the required location (a content delivery network for example)

  The source video content may be stored in a number of different locations:

  - A Viz Media Engine
  - In some other archive elsewhere on the net
  - Locally, in a location known to the Content Engine.

The plug-in also incorporates a Content Studio extension that enables Content Studio users to easily:

- Access and browse video content stored in a Viz Media Engine
- Decorate selected video content by adding template-based graphics (for example, lead in and lead out graphics, graphic overlays and so on).
- Add selected and decorated video content to suitably-configured content items
- Use Viz Media Engine to transcode video content.

This extra Content Studio functionality is only **fully** available if:

- You are running Content Studio on Microsoft Windows
- You have a correctly configured Viz Content Pilot **Newsroom Component** (from version 5.6.1) installed on your computer

If you do not have Viz Content Pilot installed on your computer or are using a different operating system, then content items that reference video objects can still be opened, but you will not be able to access Viz Media Engine to select video content.

Most of the extra functionality that the Video plug-in provides access to is provided by Vizrt video components: Viz Content Pilot, and Viz Media Engine. This functionality is therefore not described in any detail here: if you need to know more then you should consult the documentation for the relevant product.

1.1 Supported Workflows

The purpose of the Video plug-in is to enable the Content Engine to be used together with various other Vizrt systems to provide efficient production lines for online publishing of video content. All the systems involved are large and complex and can be configured in many different ways. The systems can therefore theoretically be connected together in many different way. In practice, however, this is not the case: not all configurations will work, or work well. This section contains descriptions of configurations and workflows that are known to work satisfactorily and are therefore supported by Vizrt.

All the described workflows require access to a **VME Online** system, which performs transcoding and generation of keyframe images.

1.1.1 Manual Workflows

These workflows are driven from Content Studio: the Content Studio user creates a video content item and adds a video either from Viz Media Engine or from his/her local machine.

These workflows are described in detail (from the Content Studio user's point of view) in **chapter 4**.

1.1.1.1 VME-based Workflow

In addition to **VME Online**, this workflow requires:

- Access to a **VME Video Production** system (for storage/archiving of video content)
- Access to a **Viz Content Pilot** system
- That Content Studio is being accessed from Microsoft Windows systems
- That Viz Content Pilot's **Newsroom Component** is installed on the computers used to access Content Studio.

Publishing video in this workflow consists of the following steps:

1. Content Studio user creates a video content item.
2. Content Studio user selects a video stored in VME, and adds it to the content item.
3. Content Studio user saves the content item.
4. The Content Engine sends the location of the included video content in VME to the VME Online system.
5. VME Online retrieves the video from VME.
6. VME Online transcodes the retrieved video into the required web formats.
7. VME Online updates the transcoding status of the content item in the Content Engine.
8. VME Online sends the transcoded videos to the required location (a content delivery network, for example).
9. VME Online returns the URLs of the transcoded videos, keyframes and so on to the Content Engine.
10. The Content Engine updates the content item with the returned data.

1.1.1.2 Local Upload Workflow

This workflow only requires access to a **VME Online** system. Note that this workflow is not recommended for large-scale use, since the Content Engine is not optimized for managing large volumes of video data. It is primarily intended for small scale, casual use.

Publishing video in this workflow consists of the following steps:

1. Content Studio user creates a video content item.
2. Content Studio user uploads a video from his/her local machine.
3. The Content Engine sends the URL of the uploaded video to the VME Online system
4. VME Online retrieves the video from the Content Engine.
5. VME Online transcodes the retrieved video into the required web formats.
6. VME Online updates the transcoding status of the content item in the Content Engine.
7. VME Online sends the transcoded videos to the required location (a content delivery network, for example).
8. VME Online returns the URLs of the transcoded videos, keyframes and so on to the Content Engine.
9. The Content Engine updates the content item with the returned data.

1.1.2 Automated Workflows

These workflows are driven by some external system (for example, another content management system) and do not require any interaction from Content Studio users. There are two ways of creating an automated workflow:

- Using the Content Engine web service
- Using the Content Engine syndication subsystem

The automated workflows described here are based on the assumption that the videos to be published are stored either in a VME Video Production system or in some other external system. There is no description of how to create an automated version of the local upload workflow described in **section 1.1.1.2**. This workflow is only intended for small-scale casual use and is not regarded as suitable for automation.

1.1.2.1 Web Service-based Workflow

In a web-service based workflow, the external system that is driving the publishing process communicates with the Content Engine via the Content Engine's web service, effectively emulating Content Studio.

Publishing video in this workflow consists of the following steps:

1. The external system obtains the URL of the video to be published. The video may be stored either in a **VME Video Production** system or in some other archiving system. The only requirement is that the video

archiving system makes the videos accessible to the **VME Online** system in some way. It might, for example, publish them on an HTTP, FTP or SMB server.

2. The external system creates a video content item in the Content Engine, using the Content Engine's web service. The created content item contains the URL of the video content to be published.
3. The Content Engine sends the URL of the included video content to VME Online.
4. VME Online retrieves the video from the specified location.
5. VME Online transcodes the retrieved video into the required web formats.
6. VME Online updates the transcoding status of the content item in the Content Engine.
7. VME Online sends the transcoded videos to the required location (a content delivery network, for example).
8. VME Online returns the URLs of the transcoded videos, keyframes and so on to the Content Engine.
9. The Content Engine updates the content item with the returned data.

1.1.2.2 Syndication-based Workflow

In a syndication-based workflow, the external system that is driving the publishing process communicates with the Content Engine by generating an Escenic syndication format file and starting an import process.

Publishing video in this workflow consists of the following steps:

1. The external system obtains the URL of the video to be published. The video may be stored either in a **VME Video Production** system or in some other archiving system. The only requirement is that the video archiving system makes the videos accessible to the **VME Online** system in some way. It might, for example, publish them on an HTTP, FTP or SMB server.
2. The external system creates an Escenic syndication format file that defines one or more video content items. Each content item definition contains the URL of the video content to be published.
3. The external system saves the syndication file in the Content Engine's import folder.
4. The Content Engine imports the syndication file, creating the content items defined in the syndication file.
5. The Content Engine sends the URL of the included video content to VME Online
6. VME Online retrieves the video from the specified location.
7. VME Online transcodes the retrieved video into the required web formats.
8. VME Online updates the transcoding status of the content item in the Content Engine.
9. VME Online sends the transcoded videos to the required location (a content delivery network, for example).
10. VME Online returns the URLs of the transcoded videos, keyframes and so on to the Content Engine.

11. The Content Engine updates the content item with the returned data.

2 Installation

The following preconditions must be met before you can install Video 1.2.1.162046 in a Content Engine:

- Escenic Content Engine version 5.4.2 or higher and Escenic assembly tool have been installed as described in the **Escenic Content Engine Installation Guide** and are in working order.
- You have the required plug-in distribution file **`video-dist-1.2.1.162046.zip`**.

2.1 Conventions

The instructions in the following section assume that you have a standard Content Engine installation, as described in the **Escenic Content Engine Installation Guide** for releases 5.4.2 and above. *escenic-home* is used to refer to the **`/opt/escenic`** folder under which both the Content Engine itself and all plug-ins are installed).

The Content Engine and the software it depends on may be installed on one or several host machines depending on the type of installation required. In order to unambiguously identify the machines on which various installation actions must be carried out, the **Escenic Content Engine Installation Guide** defines a set of special host names that are used throughout the manual.

Some of these names are also used here:

assembly-host
: The machine used to assemble the various Content Engine components into a enterprise archive or .EAR file.

engine-host
: The machine(s) used to host application servers and Content Engine instances.

The host names always appear in a bold typeface. If you are installing everything on one host you can, of course, ignore them: you can just do everything on the same machine. If you are creating a larger multi-host installation, then they should help ensure that you do things in the right places.

2.2 Install the Video Plug-in

Installing the Video plug-in involves the following steps:

1. Log in as **`escenic`** on your **assembly-host**.
2. Download the Video distribution from the Escenic Technet web site (http://technet.escenic.com). If you have a multi-host installation with shared folders as described in the **Escenic Content Engine Installation**

Guide, then it is a good idea to download the distribution to your shared **`/mnt/download`** folder:

```
$ cd /mnt/download
$ wget http://user:password@technet.escenic.com/downloads/55/video-dist-1.2.1.162046.zip
```

Otherwise, download it to some temporary location of your choice.

3. The Video plug-in depends on another plug-in called **VCP Editor** that manages the integration with Viz Content Pilot's Newsroom Component. You therefore need to download this plug-in's distribution file to the same location:

```
$ wget http://user:password@technet.escenic.com/downloads/55/vcpeditor-dist-??.zip
```

4. If the folder **`/opt/escenic/engine/plugins`** does not already exist, create it:

```
$ mkdir /opt/escenic/engine/plugins
```

5. Unpack the downloaded distribution files:

```
$ cd /opt/escenic/engine/plugins
$ unzip /mnt/download/video-dist-1.2.1.162046.zip
$ unzip /mnt/download/vcpeditor-dist-??.zip
```

- **`/opt/escenic/engine/plugins/video`**
- **`/opt/escenic/engine/plugins/vcpeditor`**

6. Run the **`ece`** script to re-assemble your Content Engine applications

```
$ ece assemble
```

This generates an EAR file (**`/var/cache/escenic/engine.ear`**) that you can deploy on all your **engine-host**s.

7. If you have a single-host installation, then skip this step.

On each **engine-host**, copy **`/var/cache/escenic/engine.ear`** from the **assembly-host**. If you have installed an SSH server on the **`assembly-host`** and SSH clients on your **`engine-host`**s, then you can do this as follows:

```
$ scp -r escenic@assembly-host-ip-address:/var/cache/escenic/engine.ear /var/cache/escenic/
```

where *assembly-host-ip-address* is the host name or IP address of your **assembly-host**.

8. On each **engine-host**, deploy the EAR file and restart the Content Engine by entering:

```
$ ece deploy
$ ece restart
```

2.3 Verify The Installation

To verify the status of the Video plug-in, open the Escenic Admin web application (usually located at **`http://`***server*`/escenic-admin`**) and click on

View installed plugins. The status of all currently installed plug-ins is shown here, and indicated as follows:

The plug-in is correctly installed.

The plug-in is not correctly installed.

If the Video plug-in is correctly installed, you should see something like this in the displayed plug-in list:

2.4 Update The Database Schema

The Video plug-in needs some additions to be made to the Content Engine database schema. The scripts needed to make the required additions are included in the **`misc/database/`** folder of the distribution. There are two sets of scripts, one for MySql databases, in **`misc/database/mysql`**, and one for Oracle databases in **`misc/database/oracle`**. There are three scripts in each folder:

- **`constraints.sql`**
- **`indexes.sql`**
- **`tables.sql`**

To run the scripts:

1. Log in as **`escenic`** on your **database-host**.
2. Copy or unpack the appropriate scripts for your database to an appropriate location (for example **`/tmp/video/misc/database/mysql`**).
3. Run the scripts as follows:
   - For MySql:

```
$ cd /tmp/video/misc/database/mysql/
$ for el in tables.sql indexes.sql constraints.sql; do \
  mysql -u ece-user -pece-password -h dbhost db-name < $el
  done;
```

   replacing *db-name*, *dbhost*, *ece-user* and *ece-password* with the correct values for your database.

3 Configuration

In order to be able to use the Video plug-in after installing it (see **chapter 2**) you must:

1. Create a configuration file containing all the information the Video plug-in needs to access the various systems involved in your video workflows.
2. Add one or more video content types to your publication's **content-type** resource.
3. Configure Viz Content Pilot correctly to ensure that the Content Studio browse functionality works correctly (only necessary if you are intending to use the VME-based workflow described in **section 1.1.1.1**).

While you are working on the configuration, it is a good idea to set the Content Engine logging level to **DEBUG** for the video plug-in. This will ensure that you get detailed information in the log file. Once the system is correctly configured you can set the logging level back to the default level. To change the logging level open a browser and go the **escenic-admin** application's **Logging Levels** page. Set the category **com.escenic.video** to the level you require (for example, **DEBUG**). For general information about setting logging levels, see the **Escenic Content Engine Server Administration Guide**.

3.1 Configure The Video Plug-in

The Video plug-in needs information about how to access the videos to be published and how to access the VME Online system responsible for transcoding. You provide this information by adding a configuration file to one of your configuration layers as follows:

1. Create the configuration file by copying **/opt/escenic/engine/plugins/video/misc/siteconfig/com/escenic/video/Configuration.properties** to your common configuration layer. For example:

```
$ mkdir -p /etc/escenic/engine/common/com/escenic/video/
$ cp /opt/escenic/engine/plugins/video/misc/siteconfig/com/escenic/video/
Configuration.properties \
  /etc/escenic/engine/common/com/escenic/video/Configuration.properties
```

2. Open your configuration file for editing.
3. Define values for some or all of the properties in the file. Precisely which properties need to be set depends upon the workflows your installation will need to support. See the property descriptions below for details.

The properties that may be defined in **Configuration.properties** are:

webServiceUri
The URI of your Content Engine web service, including a valid user name and password. For example:

```
webServiceUri=http://user-name:password@host/webservice/
```

where:

- *user-name* is the name of an Escenic user with access to all video content items
- *password* is the password of this user
- *host* is the Content Engine's host name or IP address

Any special characters in the URI must be URI-encoded.

This URI is used by the VME online system when transcoding to access video content items in the Content Engine. VME online needs full read/write access to these video content items. Before transcoding, it reads the URI of the actual video content to be transcoded plus any meta-information it needs from a video content item. During transcoding it updates the content item with status information. When transcoding is complete it updates the content item with the URIs of the published transcoded versions of the video, keyframes and so on.

`adactusServiceUri`

The URI of your VME Online service, including a valid user name and password. For example:

```
adactusServiceUri=http://user-name:password@host/
```

where:

- *user-name* is the name of a VME Online user
- *password* is the password of this user
- *host* is the VME Online service's host name or IP address

Any special characters in the URI must be URI-encoded. If, for example, the user name contains an @ character, this must be escaped as follows:

```
adactusServiceUri=http://post%40my-company.com:very_secret@10.211.10.8/
```

`provider`

The name of the VME Online **provider** that is to own the transcoded videos. The named provider must be defined in the VME online system. For further information about what a VME Online provider is, see the **Viz Media Engine User's Guide**.

`group`

The name of the VME Online **group** to which the transcoded videos are to belong. The named group must:

- Be defined in the VME online system.
- Belong to the specified **`provider`**.
- **Not** have a default **publish point**.

For further information about what a VME Online group is, see the **Viz Media Engine User's Guide**.

publishPoint

The number of the VME Online **publish point** at which the transcoded videos are to be published. The specified publish point must:

- Be defined in the VME online system.
- Belong to the specified **provider**.

For further information about what a VME Online group is, see the **Viz Media Engine User's Guide**.

binaryPrefix

This property only needs to be set if your system is required to support a local upload workflow (see **section 1.1.1.2**).

The video items uploaded to the Content Engine must be made accessible to the VME Online server by means of HTTP, FTP, SMB or some other network protocol. So before you can set this property you must make sure that an appropriate server is installed for serving the files to VME Online. You can then set this property to the appropriate URI.

If, for example you have set up an HTTP server to provide access to the files, you would need to specify:

```
binaryPrefix=http://host-name/
```

where *host-name* is the host name or IP address of the server you have set up.

If your system is not required to support local upload of videos, then you do not need to set this property.

---

adactusXslFile

Under normal circumstances you should not need to specify this property. If, however, the video content items retrieved by VME online do not contain the information VME Online needs to perform the required transcoding successfully, this property provides a mechanism by which the information can be modified. The property must provide the absolute URI of an XSL transformation. This transformation is then applied to every video content item retrieved by VME online. The transformation can inject additional information into every content item, or modify existing information or remove information.

externalReferencePrefix

A prefix used by VME Online when accessing the Content Engine. The default value is **escenic:**, and if the VME online system is only required to interact with one Escenic system, then you do not need to set this property. If, however, you have several parallel Escenic installations (a production system and a test or staging system, for example), all of which use the same VME online system, then you must set this property to a different value in each Escenic installation. For example:

```
externalReferencePrefix=escenic-staging:
```

ardomeFTP

This property only needs to be set if:

- Your system is required to support a VME-based workflow (see **section 1.1.1.1**) **and**
- You are using an old VME video production system called either **Ardome** or **VideoHub**.

You must specify the URL of the Ardome/VideoHub FTP server from which videos are to be retrieved. The URL must include the user name and password required to access the FTP server. For example:

```
ardomeFTP=ftp://user-name:password@host/ardome
```

where:

- *user-name* is the user name required to access the FTP server
- *password* is the password required to access the FTP server
- *host* is the host name or IP address of the machine on which the FTP server is running

---

If your video production system is based on VME ?.? or higher then you do not need to set this property.

3.2 Defining Video Content Types

In order to be able to use the Video plug-in, you need to add at least one suitably configured video content type to your publication's **content-type** resource. For general information about the **content-type** resource and how to edit it, see the **Escenic Content Engine Resource Reference**.

You can define two different video content types:

External

An external video content type can be used to hold metadata referencing video content stored either in a Viz Media Engine or in some other external server. This video content type is used for all workflows **except** the local upload workflow described in **section 1.1.1.2**.

Internal

An internal video content type includes a **link** field for holding an actual video object (an MPEG file, for example). Such content items can therefore hold video content, not just metadata and a reference. This video content type is used for the local upload workflow described in **section 1.1.1.2**.

The Video Plug-in distribution file contains an example **content-type** resource file (**video/misc/example/content-type.xml**) with more complete examples of the content type definitions described in the following sections.

3.2.1 Defining an External Video Content Type

A video content type must at least have the following:

- A parameter called **com.vizrt.video** that is set to **true**
- A **basic** field with:
  - **mime-type** set to **application/json**
  - a **video** sub-element with a **value** attribute set to **true**. This element must belong to the namespace **http://xmlns.escenic.com/2010/video**.

The following example shows such a minimal **content-type**:

```
<content-type name="external-video">
  <parameter name="com.vizrt.video" value="true"/>
  <panel name="main">
    <field name="video" type="basic" mime-type="application/json">
      <video xmlns="http://xmlns.escenic.com/2010/video" enabled="true"/>
    </field>
  </panel>
</content-type>
```

Here is a more realistic version of the same example, this time including a title field, label and so on, but with the important parts highlighted.

```
<content-type name="external-video">
  <parameter name="com.vizrt.video" value="true"/>
  <ui:icon>graphic</ui:icon>
  <ui:label>External video</ui:label>
  <ui:title-field>title</ui:title-field>
  <panel name="main">
    <field name="title" type="basic" mime-type="text/plain"/>
    <field name="video" type="basic" mime-type="application/json">
      <video xmlns="http://xmlns.escenic.com/2010/video" enabled="true"/>
    </field>
  </panel>
</content-type>
```

The **com.vizrt.video** parameter identifies the content type as a video content type managed by the Video plug-in. The video sub-element identifies its parent field as the field that is to hold video metadata. This metadata, supplied by VME Online, describes all available versions of a video object: data format, resolution, URL and so on. It is encoded in JSON format, and the parent field must therefore have its **mime-type** attribute set to **application/json**.

When a content item of this type is opened in Content Studio, then the metadata in the video field is displayed in a table, and the user can access the various versions of the video by right-clicking on a row in the table and selecting **Play** from the displayed menu.

On Windows clients only, and only if Vizrt Content Pilot is installed, a **Select video...** button is displayed above the table that can be used to select a video from Viz Media Engine. For more information about this, see **section 4.1**.

You can use external video content types to include video content from any external source, not just from Viz Media Engine. However, only Viz Media Engine video content can be accessed from Content Studio. Video

content from other sources can only be added via the web service (see **section 5.1**) or the import system (see **section 5.2**).

3.2.2 Defining an Internal Video Content Type

An internal video content type must have all the same fields and parameters as an external video type, but must in addition have a **`link`** field for holding a link to a locally stored video file. The following example shows such a content-type, with the additional **`link`** field highlighted:

```
<content-type name="internal-video">
  <parameter name="com.vizrt.video" value="true"/>
  <ui:icon>graphic</ui:icon>
  <ui:label>Internal video</ui:label>
  <ui:title-field>title</ui:title-field>
  <panel name="main">
    <field name="title" type="basic" mime-type="text/plain"/>
    <field name="binary" type="link">
      <relation>com.escenic.edit-media</relation>
    </field>
    <field name="video" type="basic" mime-type="application/json">
      <video xmlns="http://xmlns.escenic.com/2010/video" enabled="true"/>
    </field>
  </panel>
</content-type>
```

When a new content item of this type is created in Content Studio, an Open file dialog is automatically displayed so that the user can upload a suitable file to the **`link`** field. You can use **`constraint`** elements to limit the file types it is possible to upload. For example:

```
<field name="binary" type="link">
  <constraints>
    <mime-type>video/mpeg</mime-type>
    <mime-type>video/mp4</mime-type>
  </constraints>
</field>
```

You can add **`mime-type`** elements specifying any video data formats supported by VME Online. For a complete list of supported formats, see the VME Online documentation.

3.2.3 Including Key Frames in Your Video Content Items

While VME Online is ingesting and transcoding a video object, it can automatically select certain frames and save them as images called **key frames**. These key frame images can be automatically saved as content items in the Content Engine and included as relations of the video content item from which they are derived. In order to achieve this you need to add the following items to your **`content-type`** resource:

A content type for holding the key frames
This content type needs to contain a link field for storing the key frame images. A minimal key frame content type might look like this:

```
<content-type name="keyframe">
  <ui:label>Key Frame</ui:label>
  <ui:title-field>name</ui:title-field>
  <panel name="main">
    <ui:label>Image content</ui:label>
    <field name="name" type="basic" mime-type="text/plain">
```

```
      <ui:label>Name</ui:label>
      <constraints>
        <required>true</required>
      </constraints>
    </field>
    <field type="link" name="binary">
      <relation>com.escenic.edit-media</relation>
      <constraints>
        <mime-type>image/jpeg</mime-type>
        <mime-type>image/png</mime-type>
      </constraints>
    </field>
  </panel>
</content-type>
```

A relation definition

This relation definition is needed to associate key frame content items with their source video content items. It might look like this:

```
<relation-type-group name="default-relation-type-group">
  <relation-type name="keyframes">
    <ui:label>Key frame images</ui:label>
  </relation-type>
</relation-type-group>
```

Relation references

You will need to add a reference to the above relation definition to each of the video content types that you want to include key frames. For example:

```
<content-type name="video">
  <parameter name="com.vizrt.video" value="true"/>
  <ui:icon>graphic</ui:icon>
  <ui:label>External video</ui:label>
  <ui:title-field>title</ui:title-field>
  <panel name="main">
    <field name="title" type="basic" mime-type="text/plain"/>
    <field name="video" type="basic" mime-type="application/json">
      <video xmlns="http://xmlns.escenic.com/2010/video" enabled="true"/>
    </field>
    <ref-relation-type-group name="default-relation-type-group"/>
  </panel>
</content-type>
```

A `store-keyframes` element

This element causes the Video plug-in to actually store the key frames generated by VME Online as content items of the correct type. The element must belong to the namespace **`http://xmlns.escenic.com/2010/video`** and must have the following attributes:

- **`content-type`**, specifying the name of the key frame content type you have defined
- **`relation`**, specifying the name of the key frame relation you have defined
- **`state`****(optional)**, specifying the state to be applied to generated key frame content items. If not specified, the default state is **`published`**

You must add such an element to the video field in each of the video content types that you want to include key frames. For example:

```
<content-type name="video">
  <parameter name="com.vizrt.video" value="true"/>
  <ui:icon>graphic</ui:icon>
  <ui:label>External video</ui:label>
```

```
  <ui:title-field>title</ui:title-field>
  <panel name="main">
    <field name="title" type="basic" mime-type="text/plain"/>
    <field name="video" type="basic" mime-type="application/json">
      <video xmlns="http://xmlns.escenic.com/2010/video" enabled="true"/>
      <store-keyframes xmlns="http://xmlns.escenic.com/2010/video"
                       content-type="keyframe" relation="keyframes" state="draft"/>
    </field>
    <ref-relation-type-group name="default-relation-type-group"/>
  </panel>
</content-type>
```

3.3 Viz Content Pilot Configuration

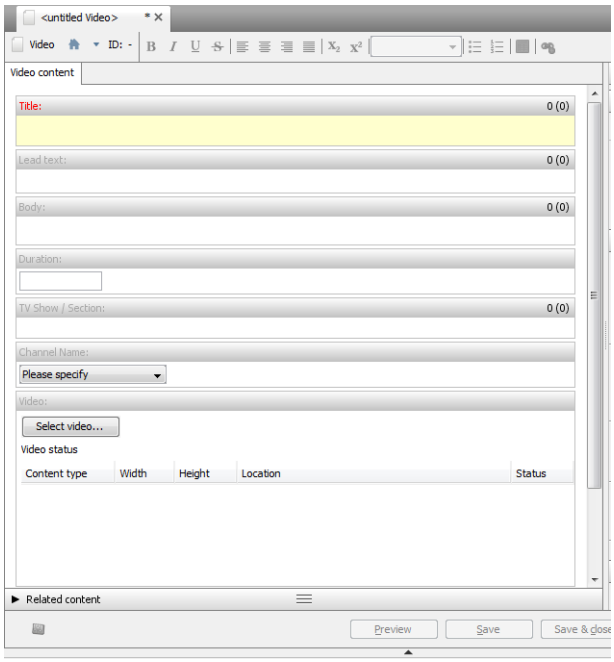
The Viz Content Pilot system's **`ax_enableMediaSendToRundown`** parameter must be set to **`y`** in order for the Content Studio browse functionality to work correctly. For details, see the the **Viz Content Pilot User's Guide**.

4 Using the Content Studio Plug-in

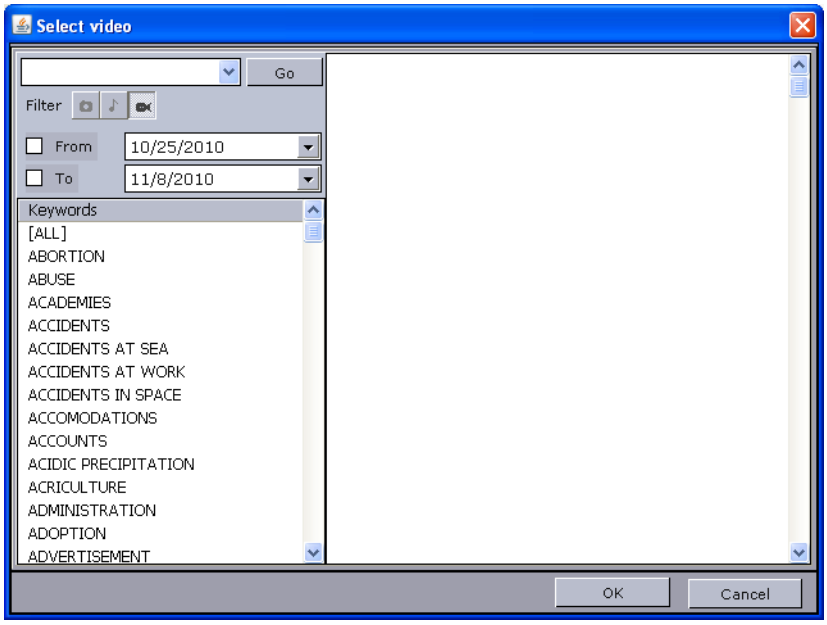
This chapter describes how to work with video content items in Content Studio on a Windows computer where Vizrt Content Pilot is installed. It is possible to open video content items on non-Windows computers or on Windows computers where Vizrt Content Pilot is not installed, but with reduced functionality.

4.1 Working With External Video Content Items

If you create a new external video content item in Content Studio, then you will see that one of the fields in the content item contains a **Select video...** button:



If you click on this button, then the following dialog is displayed:



This dialog is the **Viz Content Pilot Newsroom Component**. You can use it to:

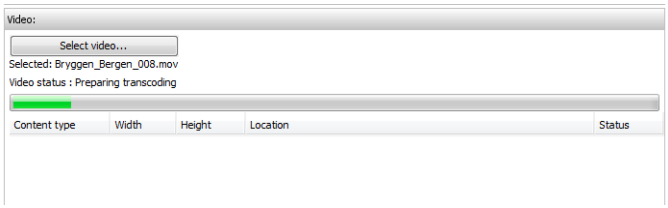
- Browse video content stored in a Viz Media Engine.
- Select the video clip you want to add to your content item.
- Decorate the selected video clip with template-based graphics.

This manual does not attempt to provide a complete description of all of the Newsroom Component's features. For a complete description of the **Newsroom Component**, see http://documentation.vizrt.com/viz-content-pilot-guide/5.6/newsroom_newsroom_integration.html.

To add a video to your content item:

1. Specify search criteria using the controls on the left side of the dialog.
2. Click on **Go**. Thumbnails of matching videos are then displayed on the right side of the dialog.
3. Pick a video. It is displayed in a new dialog, where you can play it.
4. If you want to add template-based graphics, click on **Add**. This displays a new dialog that you can use to select and fill out a graphic template.
5. When you have filled out your selected template, click on **OK**. The video is displayed again with the template graphics added. You can play the video again and adjust the timing of the graphic by dragging in the time line displayed below the video play controls.
6. You can add further template-based graphics by clicking on **Add** again.
7. When you are satisfied with the video, click **OK** to select it. The dialog then disappears and the name of the video you have selected is displayed below the **Select video...** button in the Content Studio window.

To complete the creation of the content item, click on **Save**. The video you have added is then sent to **VME Online** for transcoding. A progress bar and a sequence of status messages is displayed showing the progress of the transcoding process:

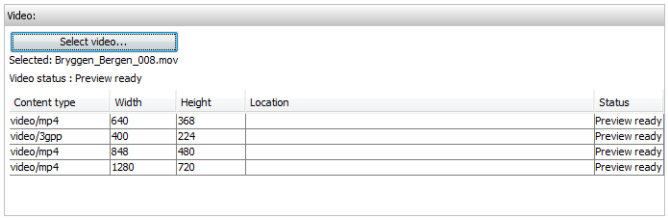


The status messages you will see are partly based on progress information sent back from VME Online. The messages correspond to VME Online states as follows:

Status message	**VME Online State**
Started	[none - the VME Online transcoding process has not yet been started]
Preparing transcoding	Downloading
Transcoding	Processing
Preview ready	Ready
Publishing	[none - VME Online has been asked to publish the transcoded videos but has not completed the operation and returned a new status]
Published	Published
Failed	Failed

For information about the precise meaning of the VME Online states, see the **Viz Media Engine Content Management Interface Guide**.

When transcoding is complete (which may take some time), details of the transcoded versions of your video are displayed in a table in the video field:



How many transcoded versions appear in the table, and what characteristics they have is determined by the set-up on your VME Online server.

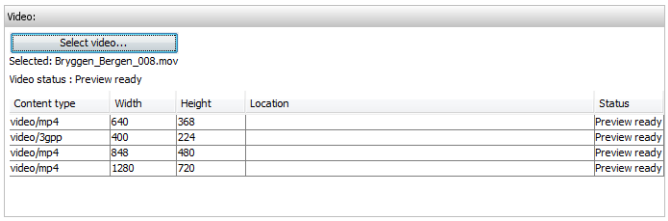
You can change the content of the video field by clicking on the **Select video...** button, selecting a new video and clicking on **Save**. The new video is then transcoded and the table is updated with the results.

If Content Pilot is not installed or if you are running Content Studio on a non-Windows computer, then the **Select video...** button is not displayed. This means that you can view the contents of existing external video content items, but you cannot change them or create new external video content items.

4.2 Working With Internal Video Content Items

If you create a new internal video content item in Content Studio, then a **File Open** dialog is immediately displayed. Depending on how the content item has been defined, this dialog will usually only allow you to select certain video file types. Select a video file to upload to the Content Engine and click **OK**. The dialog then disappears and the video file you have selected is first uploaded and then sent to VME Online for transcoding.

When transcoding is complete (which may take some time), details of the transcoded versions of your video are displayed in a table in the video field:



How many transcoded versions appear in the table, and what characteristics they have is determined by the set-up on your VME Online server.

You can change the content of the video field by clicking on the **Select video...** button again and selecting a new video. The new video is then transcoded and the table is updated with the results.

5 Automated Video Publishing

This section contains detailed information about how to implement the automated publishing workflows described in **section 1.1.2**. Whichever method you use, the basic objective is the same. You need to:

1. Define a content item of the correct type to the Content Engine. In this case it must be an external video content types as described in **section 3.2.1**. The definition must include all required fields. The most important of these is the video field described in **section 3.2.1**, which must contain JSON data specifying the URI and MIME type of the video to be published.
2. Create the defined content item. This causes the Content Engine to send the video to VME online for transcoding and keyframe generation.
3. Change the status of the content item to **`published`**.

5.1 Web Service-Based Automation

The recommended way to implement an automated video publishing process is to use the Content Engine's web service. This web service allows you to interact with the Content Engine by sending HTTP requests to the web service. For a description of the web service and how to use it general, see the Escenic Content Engine Integration Guide. The web service allows an external process to perform most of the operations that end users can perform from Content Studio. In this case the objective is to create a video content item, which involves the following steps:

1. Get the web service URI for the publication/section to which the content item is to be added (see http://documentation.vizrt.com/ece-integration-guide/5.4/navigate_the_section_hierarchy.html).
2. Make an XML document containing all the information required to create the required content item and upload it to the correct URI. This is described in general terms here: http://documentation.vizrt.com/ece-integration-guide/5.4/add_a_content_item.html.

When you are creating a video content item, the content of the VDF payload described in http://documentation.vizrt.com/ece-integration-guide/5.4/add_a_content_item.html must:

- Have a structure that matches the video content type for your publication
- Contain correctly structured JSON data in the content item's video field that specifies the specifying the URI and MIME type of the video to be published

The following example shows a VDF payload for creating a video content item, based on the content type example given in **section 3.2.1**:

```
<vdf:payload xmlns:vdf="http://www.vizrt.com/types" model="http://my-content-engine/webservice/
publication/my-publication/escenic/model/video">
  <vdf:field name="title">
    <vdf:value>My First Video</vdf:value>
  </vdf:field>
  <vdf:field name="video">
    <vdf:value>
```

```
      { "master" : { "uri" : "http://my-video-server/video1.mpeg", "mime-type" : "video/mpeg"}}
    </vdf:value>
  </vdf:field>
</vdf:payload>
```

5.2 Syndication-Based Automation

You can also implement an automated video publishing process using the Content Engine's syndication format. This is a proprietary Vizrt XML file format that can be used for import/export purposes. It is described in detail in the Escenic Content Engine Syndication Reference.

To define and create a content item in this way you simply create a syndication file containing all the required information and upload it to a specified import folder on the Content Engine server. The Content Engine will then automatically import the file and create a corresponding content item, triggering the transcoding and keyframe generation process.

For a general description of how to use the Content Engine's import service, see http://documentation.vizrt.com/ece-syndication-ref/5.4/the_import_service.html.

The following example shows a syndication file that could be used to import a video content item, based on the content type example given in **section 3.2.1**:

```
<?xml version="1.0" encoding="UTF-8"?>
<escenic xmlns="http://xmlns.escenic.com/2009/import" version="2.0">
  <content source="my-video-library" sourceid="1" type="news" state="published">
    <section-ref unique-name="videos" home-section="true"/>
    <field name="title">My First Imported Video</field>
    <field name="video">
      { "master" : { "uri" : "http://my-video-server/video1.mpeg", "mime-type" : "video/mpeg"}}
    </field>
  </content>
</escenic>
```

The video field (highlighted above) must contain the URI and the MIME type of the video, specified in JSON format, since video fields are defined with a MIME type of **`application/json`** (see **section 3.2.1**).

Although it is technically possible to import internal videos via syndication files, this is not described here. Local upload of videos to the Content Engine is primarily intended for small scale, casual use and not considered suitable for automated workflows (see **section 1.1.1.2**).

5.3 Working With the Video Field

The transcoding process that is initiated when you create a video content item results in changes to the content of the video field. All the information returned from VME Online about the transcoded variants and keyframe images that have been generated is added to the video field. In case of the example used in the previous sections, the initial information written to the video field:

```
{
```

```
  "master" : { uri" : "http://my-video-server/video1.mpeg", "mime-type" : "video/mpeg" }
}
```

might be expanded to:

```
{
  "status-uri":"http://ip-address/rest/digitalItem/status/146",
  "thumbnails":[
     {"uri":"http://ip-address/diactus/unrestricted/imageOutput/thumbs/1297_2.png","mime-
type":"image/png","width":0,"height":0},
     {"uri":"http://ip-address/diactus/unrestricted/imageOutput/thumbs/1297_3.png","mime-
type":"image/png","width":0,"height":0},
     {"uri":"http://ip-address/diactus/unrestricted/imageOutput/thumbs/1297_4.png","mime-
type":"image/png","width":0,"height":0},
     {"uri":"http://ip-address/diactus/unrestricted/imageOutput/thumbs/1297_5.png","mime-
type":"image/png","width":0,"height":0}],
  "video":[{"uri":"http://ip-address/diactus/unrestricted/published/1297.mp4","mime-type":"video/
mp4","width":0,"height":0}],
  "master" : { uri" : "http://my-video-server/video1.mpeg", "mime-type" : "video/mpeg" }
}
```

The **`status-uri`** value contains the URI of a VME Online document containing information about the transcoded variants. Your application can therefore obtain additional information about the generated variants from this file.

If the source video is stored in a Viz Media Engine then the video field will contain one additional key-value pair, called **`mos-root`**. It contains metadata about the source video in the form of an 'escaped' XML document:

```
{
 "mos-root":
    "<?xml version="\&quot;1.0\&quot;?"?>
    <!DOCTYPE mosRoot>
    <mosRoot>
      <mos>
        <mosID />
        <mosItemBrowserProgID>
        VCPAxFiller.VCPTemplateFiller&lt;\/mosItemBrowserProgID&gt;
        <mosItemEditorProgID>
        VCPAxFiller.VCPTemplateFiller&lt;\/mosItemEditorProgID&gt;
        <mosAbstract>foo&lt;\/mosAbstract&gt;
    .....
    <mosExternalMetadata>
      ....
      <mosPayload>
        ....
      </mosPayload></mosSchema></mosScope>
      </mosExternalMetadata></description></changed></createdBy></objDur></objRev></objTB></
objType></objSlug></objID></mosAbstract></mosItemEditorProgID></mosItemBrowserProgID>
    </mos>
  </mosRoot>",
  "status-uri":"http://ip-address/rest/digitalItem/status/146",
  "thumbnails":[
     {"uri":"http://ip-address/diactus/unrestricted/imageOutput/thumbs/1297_2.png","mime-
type":"image/png","width":0,"height":0},
     {"uri":"http://ip-address/diactus/unrestricted/imageOutput/thumbs/1297_3.png","mime-
type":"image/png","width":0,"height":0},
     {"uri":"http://ip-address/diactus/unrestricted/imageOutput/thumbs/1297_4.png","mime-
type":"image/png","width":0,"height":0},
     {"uri":"http://ip-address/diactus/unrestricted/imageOutput/thumbs/1297_5.png","mime-
type":"image/png","width":0,"height":0}],
  "video":[{"uri":"http://ip-address/diactus/unrestricted/published/1297.mp4","mime-type":"video/
mp4","width":0,"height":0}],
  "master" : { uri" : "http://my-video-server/video1.mpeg", "mime-type" : "video/mpeg" }
}
```

You can use the video field to store additional metadata of your own by adding add other name-value pairs to the JSON structure. You must not, however, change or remove any of the standard name-value pairs described above.

6 Accessing Video Items from Templates

You can access the transcoded versions of a video from your JSP templates using JSTL as follows (in all the examples *video-field-name* is the name of the video field in your content type and *index* is the index of the transcoded version you want):

- To access the URI of a transcoded video stored on the VME Online server:

  ```
  ${article.fields.video-field-name.value.video[index].uri}
  ```

- To access the video's MIME type:

  ```
  ${article.fields.video-field-name.value.video[index].mime-type}
  ```

- To access the video's height:

  ```
  ${article.fields.video-field-name.value.video[index].height}
  ```

- To access the video's width:

  ```
  ${article.fields.video-field-name.value.video[index].width}
  ```