Video

Table of Contents

Extends: Group

Description

The **Video** node class provides a controlled play of live or VOD video.

The **Video** node includes a wide variety of internal nodes to support trick play, playback buffering indicators, and so forth. Playback buffering indicators, to indicate buffering before initial playback as well as re-buffering, use an internal instance of a **ProgressBar** node. For trick play, an internal instance of a **TrickPlayBar** node is provided. For display of BIF images for DVD-like chapter selection, an internal instance of a **BIFDisplay** node is provided.

Starting from firmware 8, the behavior of the Roku system overlay is such that the system overlay now slides in whenever the * button is pressed, the Video node is in focus, and the app does not have its OnKeyEvent() handler fired. When the Video node is not in focus, the system overlay does not slide in and the OnKeyEvent() handler is fired.

Field Types

Playback Fields

To set the specific video playback parameters for a particular video, set the **Content Meta-Data** attributes for the video in a **ContentNode** node, and assign the **ContentNode** node to the `content` field of the **Video** node.

Video playback can then be controlled by setting the value of the `control` field, such as setting the field value to `play` to begin playback.

The `control` field includes a `prebuffer` option, which allows the video to begin buffering without showing the video. You can use this option to begin buffering of a video before a user has actually selected and started the video, such as when the user is looking at information about various video offerings in a list or grid or another type of UI element. This can eliminate much or all of the apparent delay in starting the video due to buffering the video for the user. For example, you could set the `control` field value to `prebuffer` in a callback function triggered by the `itemFocused` events that occur as a user scrolls down a list of video offerings that also display information about each video. When the user makes the selection, you can begin the actual video playback by setting the `control` field value to `play` in a callback function triggered by the `itemSelected` event for the list.

Field	Type	Default	Use

---

content	**ContentNode**	Invalid	The **ContentNode** node with the **Content Meta-Data** for the video, or a video playlist (a sequence of videos) to be played. If a video playlist is to be played, the children of this **ContentNode** node comprise the playlist, and each **Content Node** child must have all attributes required to play that video. For example, if the videos "A" and "B" are to be played, three **ContentNode** nodes must be created: the parent **ContentNode** (which is largely ignored), one **ContentNode** child for "A," and one **ContentNode** child for "B." The parent node is set into this `content` field, and when video playback is started, all of its children will be played in sequence. Any changes made to the playlist after playback has started are ignored. See the `contentIsPlaylist` and `contentIndex` fields, for more information on playlists.
contentIsPlaylist	Boolean	false	If set to true, enables video playlists (a sequence of videos to be played). See the `content` and `contentIndex` field for more information on playlists. *Available since firmware version 7.2*
contentIndex	integer	-1	**Read-Only** The index of the video in the video playlist that is currently playing. Generally, you would only want to check this field if video playlists are enabled (by setting the `contentIsPlaylist` field to true), but it is set to 0 when a single video is playing, and video playlists are not enabled. *Available since firmware version 7.2*
nextContentIndex	integer	-1	If the `contentIsPlaylist` field is set to true to enable video playlists, sets the index of the next video in the playlist to be played. Setting this field does not immediately change the video being played, but takes effect when the current video is completed or skipped. By default, this value is -1, which performs the default index increment operation. After the video specified by the index in this field begins playing, the field is set to the default -1 again, so the next video played will be set by the default index increment operation unless the field is set again to a different index. *Available since firmware version 7.2*
control	option string	"none"	Sets the desired play state for the video, such as starting or stopping the video play. Getting the value of this field returns the most recent value set, or `none` if no value has been set. To dynamically monitor the actual state of the video, see the `state` field. The play and stop commands to commence and discontinue playback should not be used to implement trick modes like rewind, or replay. For that use the **seek** field. **Option** / **Effect** none: No play state set play: Start video play stop: Stop video play pause: Pause video play resume: Resume video play after a pause replay: Replay video prebuffer: Starts buffering the video stream before the **Video** node actually begins playback. Only one video stream can be buffering in the application at any time. Setting the `control` field to `prebuffer` for another video stream after setting `prebuffer` for a previous video stream stops the buffering of the previous video stream. *Available since firmware version 7.2* skipcontent: Skip the currently-playing content and begin playing the next content in the playlist. If the content is not a playlist, or if the current content is the end of the playlist, this will end playback. *Available since firmware version 7.2*

<table>
<tr><td>state</td><td>value string</td><td>"none"</td><td>Read-Only
Describes the current video play state, such as if the video play has been paused.
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>none</td><td>No current play state</td></tr><tr><td>buffering</td><td>Video stream is currently buffering</td></tr><tr><td>playing</td><td>Video is currently playing</td></tr><tr><td>paused</td><td>Video is currently paused</td></tr><tr><td>stopped</td><td>Video is currently stopped</td></tr><tr><td>finished</td><td>Video has successfully completed playback</td></tr><tr><td>error</td><td>An error has occurred in the video play. The error code and error message can be found in the <code>errorCode</code> and <code>errorMsg</code> fields respectively.</td></tr></table></td></tr>
<tr><td>errorCode</td><td>integer</td><td>0</td><td>Read-Only
The error code associated with the video play error set in the <code>state</code> field.</td></tr>
<tr><td>errorMsg</td><td>string</td><td>""</td><td>Read-Only
An error message describing the video play error set in the <code>state</code> field.</td></tr>
</table>

Trickplay Fields

Field	Type	Default	Use
duration	Double	0	**Read-Only** The duration of the video being played, specified in seconds. This becomes valid when playback begins and may change if the video is dynamic content, such as a live event.
loop	Boolean	false	If set to true, the video or video playlist (if the `contentIsPlaylist` field is set to true to enable video playlists) will be restarted from the beginning after the end is reached. *Available since firmware version 7.2*
position	Double	0	**Read-Only** Time of the current position in the stream. Either UTC time or elapsed since start of stream depending on content type
notificationInterval	time	0.5	The interval between notifications to observers of the position field, specified as the number of seconds. If the value is 0, no notifications are delivered. This value may be read or modified at any time.
seek	time	invalid	**Write-Only** Sets the current position in the video. The value is the number seconds from the beginning of the stream, specified as a double.
timedMetaData	associative array	{ }	**Read-Only** The most recent timed meta data that has been decoded from the video stream. Only meta data with a key that matches an entry in timedMetaDataSelectionKeys will be set into this field. The value of this field is an associative array which contains arbitrary keys and values, as found in the video stream.
timedMetaDataSelectionKeys	array of strings	[ ]	If the video stream contains timed meta data such as ID3 tags, any meta data with a key matching an entry in this array will be set into the timedMetaData field. If any entry in this array is "*", then all timed meta data will be selected.

<table>
<tr><td>streamInfo</td><td>associative array</td><td>invalid</td><td>Read-Only
Information about the video stream that is currently playing or buffering.
<table><tr><th>Key</th><th>Type</th><th>Value</th></tr><tr><td>isUnderrun</td><td>Boolean</td><td>If true, the stream was downloaded due to an underrun</td></tr><tr><td>isResumed</td><td>Boolean</td><td>If true, playback was resumed after trickplay</td></tr><tr><td>measuredBItrate</td><td>Integer</td><td>The measured bitrate (bps) of the network when the stream was selected</td></tr><tr><td>streamBitrate</td><td>Integer</td><td>The bitrate of the stream</td></tr><tr><td>streamUrl</td><td>URI</td><td>The URL of the stream</td></tr></table>
Available since firmware version 7.2</td></tr>
<tr><td>completedStreamInfo</td><td>associative array</td><td>invalid</td><td>Read-Only
Information about the video stream that most recently completed playing, due to an error, user action, or end of the stream. The associative array consists of the same keys as for the <code>streaminfo</code> field, with one additional key, <code>isFullResult</code>, a <code>Boolean</code> type that, if true indicates the stream played to completion, or if false, was interrupted by an error or user action. This field is set prior to the <code>state</code> field being changed, so <code>state</code> field observer callback functions can assume that the associative array values are valid when the <code>state</code> field changes.
Available since firmware version 7.2</td></tr>
<tr><td>timeToStartStreaming</td><td>Double</td><td>0</td><td>Read-Only
The time in milliseconds from playback being started until the video actually began playing. The minimum valid value is 1 millisecond, and this is only valid if the current value of the <code>state</code> field is <code>playing</code>. When the state field value is not <code>playing</code>, the value will be 0. This field is updated prior to the <code>state</code> field changing, so <code>state</code> field observer callback functions can assume this field is valid after the <code>state</code> field value changes to <code>playing</code>.
Available since firmware version 7.2</td></tr>
<tr><td>bufferingStatus</td><td>associative array</td><td>invalid</td><td>Read-Only
Contains information about stream buffering progress and status. This field is valid only while buffering is in progress, both at stream startup or when re-buffering is required. Observers will be notified when any element of the array changes, and also when buffering is complete and the field itself becomes invalid. The array contains the following name - value pairs.
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>percentage</td><td>Percent buffering complete as an integer.</td></tr><tr><td>isUnderrun</td><td>Boolean value indicating if a stream underrun occurred.</td></tr></table></td></tr>
</table>

<table>
<tr><td>videoFormat</td><td>string</td><td>""</td><td>Read-Only
Contains the format of the currently playing video stream.
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>""</td><td>No stream playing</td></tr><tr><td>none</td><td>Stream contains no playable video</td></tr><tr><td>unknown</td><td>Stream contains unknown video</td></tr><tr><td>hevc</td><td>ISO/IEC 23008-2, H.265, HEVC</td></tr><tr><td>hevc_b</td><td>ISO/IEC 23008-2 Annex-B, H.265, HEVC</td></tr><tr><td>mpeg1</td><td>ISO/IEC 11172-2, MPEG-1 part 2, H.261</td></tr><tr><td>mpeg2</td><td>ISO/IEC 13818-2, MPEG-2 part 2, H.262</td></tr><tr><td>mpeg4_2</td><td>ISO/IEC 14496-2, MPEG-4 part 2, H.263</td></tr><tr><td>mpeg4_10b</td><td>ISO/IEC 14496-10, MPEG-4 part 10 Annex-B, H.264, vc-1</td></tr><tr><td>mpeg4_15</td><td>ISO/IEC 14496-15, MPEG-4 part 15, H.264, vc-1</td></tr><tr><td>AVC
vc1</td><td>vc-1</td></tr><tr><td>wmv</td><td>Microsoft Windows Media Video</td></tr><tr><td>vp8</td><td>VP8 codec</td></tr><tr><td>vp9</td><td>VP9 codec</td></tr></table></td></tr>
<tr><td>pauseBufferStart</td><td>Double</td><td>0</td><td>Read-Only
The beginning position of the video buffered when paused. This field is only valid for live video.</td></tr>
<tr><td>pauseBufferEnd</td><td>Double</td><td>0</td><td>Read-Only
The ending position of the video buffered when paused. This field is only valid for live video.</td></tr>
<tr><td>pauseBufferOverflow</td><td>Boolean</td><td>false</td><td>Read-Only
Indicates that the video buffer was not able to save all video since being paused. This field is only valid for live video.</td></tr>
<tr><td>streamingSegment</td><td>associative array</td><td>{ }</td><td>Read-Only
Information about the video segment that is currently streaming. This is only meaningful for segmented video transports, such as DASH and HLS. The associative array has the following entries:
<table><tr><th>Key</th><th>Type</th><th>Value</th></tr><tr><td>segBitrateBps</td><td>integer</td><td>Bitrate of the segment in bits per second</td></tr><tr><td>segSequence</td><td>integer</td><td>The sequence number of the segment in the video</td></tr><tr><td>segStartTime</td><td>time</td><td>The start time of the segment from the start of the video, specified in seconds</td></tr><tr><td>segUrl</td><td>string</td><td>URL of the segment</td></tr></table></td></tr>
</table>

<table>
<tr>
<td>downloadedSegment</td>
<td>associative array</td>
<td>invalid</td>
<td>Read-Only
Information about the video segment that was just downloaded. This is only meaningful for segmented video transports, such as DASH and HLS. The associative array has the following entries:
<table>
<tr><th>Key</th><th>Type</th><th>Value</th></tr>
<tr><td>Status</td><td>integer</td><td>Status of the download: 0 = success, nonzero = error</td></tr>
<tr><td>Sequence</td><td>integer</td><td>Stream segment sequence number</td></tr>
<tr><td>SegUrl</td><td>string</td><td>Stream segment URL (i.e., .ts file for HLS, stream fragment URL for smooth)</td></tr>
<tr><td>DownloadDuration</td><td>integer</td><td>Amount of time spent downloading the segment, in milliseconds</td></tr>
<tr><td>SegSize</td><td>integer</td><td>Segment size, in bytes</td></tr>
<tr><td>SegType</td><td>integer</td><td>Type of data in the segment: 1=audio, 2=video, 3=captions, 0=mux</td></tr>
<tr><td>BitrateBPS</td><td>integer</td><td>Bitrate of the segment, in bits per second</td></tr>
</table>
</td>
</tr>
</table>

manifestData	associative array	{ }	*This function is available in firmware 7.7 or later.*
			
			"manifestData" detect the periods in a DASH manifest before they are played back. One major use case for this is to display ad markers in the trickplay progress bar.
			
			The manifestData field has two elements:
			
			• "mpd" — `roAssociativeArray` of string values
			• "periods" — `roArray` of `roAssociativeArrays` of string values
			
			The property `minimumUpdatePeriod` has also been added to control the .mpd element.
			
			The “periods” element includes a Period key for each period in the manifest, with a value of attributes in the Period key. For example, a period might contain the following values:
			{ id="p24895847", start="PT1492010820S", duration="PT60S" }
			
			Some examples of how to access the manifestData would include:
			
			1. Get a known attribute:
			`video.manifestData.mpd.minimumUpdatePeriod`
			
			2. Get a known attribute which has a semicolon in the name:
			`video.manifestData.mpd["xmlns:ns1"]`
			
			3. Get a known attribute from existing period:
			`video.manifestData.period[0].id`
			
			4. Get number of available periods:
			`video.manifestData.periods.Count()`
			
			5. Iterate through all available MPD attributes:
			`for each item in video.manifestData.mpd.Items()`
			`print item.key, "=", item.value`
			`end for`

UI Fields

Field	Type	Default	Use
width	float	0.0	Sets the width of the video play window in pixels. If set to 0.0 (the default), the video play window is set to the width of the entire display screen.
height	float	0.0	Sets the height of the video play window in pixels. If set to 0.0 (the default), the video play window is set to the height of the entire display screen.
enableUI	Boolean	true	If set to true (the default), the entire **Video** node user interface (such as **ProgressBar** and **TrickPlayBar** nodes, and BIF navigation) appear in response to stream events and remote control key presses. If set to false, most of the **Video** node user interface will not be shown, and the application is expected to implement the UI. The one exception is the closed-caption dialog, which always appears when the video is playing fullscreen and the user presses the * (Info) button When using the Roku Advertising Framework (RAF), the RAF library may temporarily set this field to false while playing ads. *Available since firmware version 7.2*
enableTrickPlay	Boolean	true	Controls whether trickplay is allowed during playback. When set to false the user trickplay buttons on the remote will have no effect. This only applies when enableUI is set to true.

<table>
<tr>
<td>bifDisplay</td>
<td>BifDisplay node</td>
<td>internal instance default</td>
<td>Component that displays BIFs and allows navigation. The fields of this internal node are as follows:
<table>
<tr><th>Field</th><th>Type</th><th>Default</th><th>Use</th></tr>
<tr><td>frameBgBlendColor</td><td>color</td><td>0xFFFFFFFF</td><td>A color to be blended with the image displayed behind individual BIF images displayed on the screen. The blending is performed by multiplying this value with each pixel in the image. If not changed from the default value, no blending will take place.</td></tr>
<tr><td>frameBgImageUri</td><td>uri</td><td>""</td><td>The URI of an image to be displayed behind individual frames on the screen. The actual frame image is displayed opaquely on top of this background, so only the outer edges of this image are visible. Because of that, this background image typically appears as a border around the video frame. If the <code>frameBgBlendColor</code> field is set to a value other than the default, that color will be blended with the background image.</td></tr>
<tr><td>getNearestFrame</td><td>time</td><td>invalid</td><td>Available since firmware version 9

Write-Only Requests the nearest BIF to the time specified. This would normally be an offset from the current playback position. The <code>getNearestFrame</code> request is passed to the BifCache which uses the <code>getNearestFrame()</code> method implemented on all BIF storage classes. Existing BifCache functionality is then used to retrieve the bitmap data and load it into the texture manager.</td></tr>
<tr><td>nearestFrame</td><td>string</td><td>""</td><td>Available since firmware version 9

Read-Only Contains the URI of the requested BIF. The returned URIs will be of the form 'memory://BIF_%d_%d'. These URIs can then be used directly in the 'uri' field of a Poster SGN (or similar).</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>trickPlayBar</td>
<td>TrickPlayBar node</td>
<td>internal instance default</td>
<td>The visible TrickPlayBar node. The fields of this internal node are as follows:
<table>
<tr><th>Field</th><th>Type</th><th>Default</th><th>Use</th></tr>
<tr><td>currentTimeMarkerBlendColor</td><td>color</td><td>0xFFFFFFFF</td><td>This is blended with the marker for the current playback position. This is typically a small vertical bar displayed in the TrickPlay Bar node when the user is fast-forwarding or rewinding through the video.</td></tr>
<tr><td>textColor</td><td>color</td><td>system default</td><td>Sets the color of the text next to the trickPlayBar node indicating the time elapsed/remaining.
Available since firmware version 7.2</td></tr>
<tr><td>thumbBlendColor</td><td>color</td><td>0xFFFFFFFF</td><td>Sets the blend color of the square image in the trickPlayBar node that shows the current position, with the current direction arrows or pause icon on top. The blending is performed by multiplying this value with each pixel in the image. If not changed from the default value, no blending will take place.
Available since firmware version 7.2</td></tr>
<tr><td>filledBarBlendColor</td><td>color</td><td>0xFFFFFFFF</td><td>This color will be blended with the graphical image specified in the <code>filledBarImageUri</code> field. The blending is performed by multiplying this value with each pixel in the image. If not changed from the default value, no blending will take place.
Available since firmware version 7.2</td></tr>
<tr><td>filledBarImageUri</td><td>uri</td><td>""</td><td>A 9-patch or ordinary PNG of the bar that represents the completed portion of the work represented by this ProgressBar node. This is typically displayed on the left side of the track. This will be blended with the color specified by the <code>filledBarBlendColor</code> field, if set to a non-default value.
Available since firmware version 7.2</td></tr>
<tr><td>trackBlendColor</td><td>color</td><td>0xFFFFFFFF</td><td>This color is blended with the graphical image specified by <code>trackImageUri</code> field. The blending is performed by multiplying this value with each pixel in the image. If not changed from the default value, no blending will take place.
Available since firmware version 7.2</td></tr>
<tr><td>trackImageUri</td><td>uri</td><td>""</td><td>A 9-patch or ordinary PNG of the track of the progress bar, which surrounds the filled and empty bars. This will be blended with the color specified by the <code>trackBlendColor</code> field, if set to a non-default value.
Available since firmware version 7.2</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>bufferingBar</td>
<td>ProgressBar node</td>
<td>internal instance default</td>
<td>Component that shows the progress of re-buffering, after video playback has started. The fields of this internal node are as follows:
<table>
<tr><th>Field</th><th>Type</th><th>Default</th><th>Use</th></tr>
<tr><td>width</td><td>float</td><td>system default</td><td>Sets a custom width for an instance of the Progress Bar node
Available since firmware version 7.2</td></tr>
<tr><td>height</td><td>float</td><td>system default</td><td>Sets a custom height for an instance of the Progress Bar node
Available since firmware version 7.2</td></tr>
<tr><td>emptyBarBlendColor</td><td>color</td><td>0xFFFFFFFF</td><td>A color to be blended with the graphical image specified in the <code>emptyBarImageUri</code> field. The blending is performed by multiplying this value with each pixel in the image. If not changed from the default value, no blending will take place.</td></tr>
<tr><td>emptyBarImageUri</td><td>uri</td><td>""</td><td>A 9-patch or ordinary PNG of the bar presenting the remaining work to be done. This is typically displayed on the right side of the track, and is blended with the color specified in the <code>emptyBarBlendColor</code> field, if set to a non-default value.</td></tr>
<tr><td>filledBarBlendColor</td><td>color</td><td>0xFFFFFFFF</td><td>This color will be blended with the graphical image specified in the <code>filledBarImageUri</code> field. The blending is performed by multiplying this value with each pixel in the image. If not changed from the default value, no blending will take place.</td></tr>
<tr><td>filledBarImageUri</td><td>uri</td><td>""</td><td>A 9-patch or ordinary PNG of the bar that represents the completed portion of the work represented by this ProgressBar node. This is typically displayed on the left side of the track. This will be blended with the color specified by the <code>filledBarBlendColor</code> field, if set to a non-default value.</td></tr>
<tr><td>trackBlendColor</td><td>color</td><td>0xFFFFFFFF</td><td>This color is blended with the graphical image specified by <code>trackImageUri</code> field. The blending is performed by multiplying this value with each pixel in the image. If not changed from the default value, no blending will take place.</td></tr>
<tr><td>trackImageUri</td><td>uri</td><td>""</td><td>A 9-patch or ordinary PNG of the track of the progress bar, which surrounds the filled and empty bars. This will be blended with the color specified by the <code>trackBlendColor</code> field, if set to a non-default value.</td></tr>
<tr><td>percentage</td><td>integer</td><td>top</td><td>The percentage of the work that is done. Setting this field controls the visual appearance of the ProgressBar node.</td></tr>
</table>
</td>
</tr>
<tr>
<td>bufferingTextColor</td>
<td>color</td>
<td>system default</td>
<td>The color of the text displayed near the buffering bar defined by the <code>bufferingBar</code> field, when the buffering bar is visible. If this is 0, the system default color is used. To set a custom color, set this field to a value other than 0x0.</td>
</tr>
<tr>
<td>retrievingBar</td>
<td>ProgressBar node</td>
<td>internal instance default</td>
<td>Component that shows the progress of initial retrieving of the video, prior to starting playback. The fields of this internal node are the same as for the <code>bufferingBar</code> field, which are the fields of the internal ProgressBar node.</td>
</tr>
<tr>
<td>retrievingTextColor</td>
<td>color</td>
<td>system default</td>
<td>The color of the text displayed near the retrieving bar, when the retrieving bar defined in the <code>retrievingBar</code> field is visible. If this is 0, the system default color is used. To set a custom color, set this field to a value other than 0x0.</td>
</tr>
</table>

Closed Caption Fields

<table>
<tr><th>Field</th><th>Type</th><th>Default</th><th>Use</th></tr>
<tr><td>globalCaptionMode</td><td>option string</td><td>"Off"</td><td>Sets the value of the global Roku closed-caption mode. This can be used to allow the user or the application to change the closed-caption mode in an application just before or during video playback. The possible options are:

<table><tr><th>Option</th><th>Effect</th></tr><tr><td>"Off"</td><td>Turns the global Roku closed-caption mode off.</td></tr><tr><td>"On"</td><td>Turns the global Roku closed-caption mode on.</td></tr><tr><td>"Instant replay"</td><td>Sets the global Roku closed-caption setting to display captions only during instant replay.</td></tr><tr><td>"When mute"</td><td>Sets the global Roku closed-caption setting to display captions only when the volume is muted. (This only applies to Roku TVs.)</td></tr></table>
Orthogonal functionality - The channel should set the subtitleTrack regardless of the selected Caption Mode.</td></tr>
<tr><td>suppressCaptions</td><td>boolean</td><td>false</td><td>Suppresses the closed caption for the purpose of resolving conflicts in cases where UI elements are drawn.

Note that most of the disabling/enabling of the captions are done by the firmware, including enabling closed caption for Instant Replay.</td></tr>
<tr><td>subtitleTrack</td><td>string</td><td>""</td><td>The identifier of the selected subtitle track. Subtitles may or may not be visible on the screen, depending upon the user's caption mode setting. Reading this field will return the identifier of the current subtitle track, and setting the field will change the track.

Subtitle track should always be set regardless of the mode.

See also: globalCaptionMode</td></tr>
<tr><td>availableSubtitleTracks</td><td>array of associative arrays</td><td>[ ] empty array</td><td>Read-Only
The list of subtitle tracks available in the video stream. The array is initially populated with the tracks specified in the Content Meta-Data, and additional tracks are added if they are detected by the digital video player. Each associative array has the following entries:

<table><tr><th>Key</th><th>Type</th><th>Value</th></tr><tr><td>Description</td><td>string</td><td>Descriptive name of the subtitle track</td></tr><tr><td>Language</td><td>string</td><td>ISO 639-2 three-letter language code</td></tr><tr><td>TrackName</td><td>string</td><td>The track identifier. The value of this field may be used to select the subtitle track.</td></tr></table></td></tr>
<tr><td>captionStyle</td><td>Associative array</td><td>System default</td><td>Allows channels to style closed captions. For any keys that are absent from the associative array, or for unexpected values, the Default value is assumed for that property. Following are the possible key names and values for this field:

<table><tr><th>Property</th><th>Possible Values</th></tr></table></td></tr>
</table>

Text style	Default Serif Fixed Width Serif Proportional Sans Serif Fixed Width Sans Serif Proportional Casual Cursive Small Caps
Text edge effect	Default None Raised Depressed Uniform Drop shadow (left) Drop shadow (right)
Text size	Default Large Medium Small
Text color	Default White Black Red Green Blue Yellow Magenta Cyan
Text opacity	Default 25% 75% 100%
Background Color	Default White Black Red Green Blue Yellow Magenta Cyan
Background Opacity	Default Off 25% 75% 100%
Window Color	Default White Black Red Green Blue Yellow Magenta Cyan

			Window Opacity: Default, Off, 25%, 75%, 100% *Available since firmware version 8.*

Audio Fields

Field	Type	Default	Use
mute	Boolean	false	Set to true to mute the audio of the video currently playing in the **Video** node. Set to false to restore audio.
audioTrack	string	""	The track identifier of the audio being played. Reading this field will return the track identifier of the audio being played, and writing this value will change the audio track.
availableAudioTracks	array of associative arrays	[ ] *empty array*	**Read-Only** Each associative array has the following entries: **Key** / **Type** / **Value** Language / string / ISO 639-2 three-letter language code Name / string / Descriptive name of the audio track Track / string / The track identifier. The value of this field may be used to select the audio track. The field also retrieves audio description tracks which are typically seen on broadcast TV. An audio description track is mixed with the main audio track.

<table>
<tr><td>audioFormat</td><td>string</td><td>""</td><td>Read-Only
Contains the format of the currently playing audio.
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>""</td><td>No stream playing</td></tr><tr><td>none</td><td>Stream contains no playable audio</td></tr><tr><td>unknown</td><td>Stream contains unknown audio</td></tr><tr><td>aac</td><td>ISO/IEC 14496-3, Advanced Audio Coding</td></tr><tr><td>aac_adif</td><td>ISO/IEC 14496-3, Advanced Audio Coding, ADIF container</td></tr><tr><td>aac_adts</td><td>ISO/IEC 14496-3, Advanced Audio Coding, ADTS container</td></tr><tr><td>aac_latm</td><td>ISO/IEC 14496-3, Advanced Audio Coding, LATM container</td></tr><tr><td>ac3</td><td>Dolby Digital</td></tr><tr><td>alac</td><td>Apple Lossless</td></tr><tr><td>dts</td><td>DTS Coherent Acoustics</td></tr><tr><td>eac3</td><td>Dolby Digital Plus</td></tr><tr><td>flac</td><td>Free Lossless Audio Codec</td></tr><tr><td>mp2</td><td>ISO/IEC 11172-3, MPEG Audio Layer II</td></tr><tr><td>mp3</td><td>ISO/IEC 11172-3, MPEG Audio Layer III</td></tr><tr><td>pcm</td><td>linear PCM</td></tr><tr><td>vorbis</td><td>Ogg Vorbis</td></tr><tr><td>wma</td><td>Microsoft Windows Media Audio</td></tr><tr><td>wmapro</td><td>Microsoft Windows Media Pro Audio</td></tr></table></td></tr>
<tr><td>supplementaryAudioVolume</td><td>int</td><td>50</td><td>This field sets the volume of the description tracks separately from the main audio track. The field value can range from 0 to 100.

Available since firmware version 8.</td></tr>
</table>

Miscellaneous Fields

Field	Type	Default	Use

MaxVideoDecodeResolution	vector2d (width, height)	[0,0]	Sets the max resolution required by your video. Video decode memory is a shared resource with OpenGL texture memory. The Brightscript 2D APIs are implemented using OpenGL texture memory on Roku models that support the Open GL APIs (please see Roku Models and Features for a list of these models). On models that do not support Open GL APIs, this field exists for API compatibility but has no effect on actual memory allocations. Video decode memory allocation is based on a resolution of 1920x1080 or 1280x720 as the maximum supported resolution for a particular Roku model (please see Roku Models and Features for a list of these models). This field enables applications that want to use both the 2D APIs and video playback with a lower resolution than 1080p. Without this field, these applications are likely to not have enough memory for either video playback or UI rendering. If width is 0 (the default), it is unlimited. If height is 0 (the default), it is unlimited. *Available since firmware version 7.2*
cgms	integer	0	Sets the CGMS (Copy Guard Management System) on analog outputs to the desired level. The valid values are: **Value** / **Effect** 0 — No copy restriction 1 — Copy no more 2 — Copy once allowed 3 — No copying permitted
enableScreenSaverWhilePlaying	Boolean	false	Set this to true to allow the screensaver to activate even if video is playing, as long as that video does not cover 50% or more of the screen. Set to false to block the screensaver activating if any video is playing. Note that this field has no effect when the video node plays audio only streams. For screensaver control with audio only streams, use the disableScreenSaver field.
disableScreenSaver	Boolean	false	Set this to true to suppress the screensaver. This is typically used to suppress the screensaver when playing audio-only streams.
contentBlocked	Boolean	false	**Read-Only** Determines whether the current content is blocked. *Available since firmware version 8.*

Data Bindings

See **Content Meta-Data** for the required and optional play parameters, and descriptive information for video playback. Set these parameters in a **ContentNode** node, and assign the **ContentNode** node to the `content` field of the **Video** node to apply the parameters to a particular video content item.

For HTTPS access, note the following **Content Meta-Data** attributes:

- `HttpCertificatesFile`
- `HttpCookies`
- `HttpHeaders`
- `HttpSendClientCertificates`

These attributes must be set to handle secure HTTP transfers of video files. Note that this is a different HTTPS mechanism than used for other

SceneGraph nodes as described in roHttpAgent.

> In firmware versions prior to 7.2, each **Audio** and **Video** node created and configured an `HttpAgent` only when the first content was played in a given **Audio** or **Video** node instance. This sometimes meant that additional content would fail to play in the same node because headers, cookies, and certificates were not updated or correctly replaced from the new content record. Channels that are dependent upon this behavior will need to be updated to set the required data into the **Content Meta-Data** for each piece of content, or to programmatically set those values into the `HttpAgent` before playing each piece of content.

Example

To play video in an application, you first need to create a **Video** node, either in BrightScript using the **roSGNode** **ifSGNodeChildren** interface, or in XML markup. For example, in XML markup:

```
<Video
  id="musicvideos"
  width="1280"
  height="720"
  translation="[0,0]"
/>
```

The **Video** node is then scripted to specify the URL of the video stream, streaming format, video title, and any other **Content Meta-Data** attributes needed for the particular playback. Once the video properties are specified, the video can be played by setting the **Video** node `control` field value to `play`.

```
<script type="text/brightscript" >
<![CDATA[

sub init()
  m.top.setFocus(true)
  setVideo()
end sub

function setVideo() as void
  videoContent = createObject("RoSGNode", "ContentNode")
  videoContent.url =
"https://roku.s.cpl.delvenetworks.com/media/59021fabe3b645968e382ac726cd6c7b/60b4a471ffb
74809beb2f7d5a15b3193/roku_ep_111_segment_1_final-cc_mix_033015-a7ec8a288c4bcec001c11818
1c668de321108861.m3u8"
  videoContent.title = "Test Video"
  videoContent.streamformat = "hls"

  m.video = m.top.findNode("musicvideos")
  m.video.content = videoContent
  m.video.control = "play"
end function

]]>
</script>
```