MPCDI

ON THIS PAGE

- Support Overview
  - Player Models
  - Media Types
- BrightScript API for MPCDI
  - Enabling MPCDI
    - SetMpcdi(parameters As roAssociativeArray) As Boolean
  - Disabling MPCDI
- MPCDI Template Script
- Setting MPCDI Regions
  - BrightSign Shell
  - BrightScript

BrightSign players support the MPCDI specification for warping and edge blending. This page outlines how to display media—including images, video, and HTML—using MPCDI. Note that some knowledge of BrightScript is required to develop MPCDI applications on BrightSign players.

Note

The API described below requires firmware version 7.0 or later.

Support Overview

PLAYER MODELS

Series 4 (XTx44, XDx34, HDx24) and Series 3 (XTx43, XDx33, HDx23, HO523) models support all levels of the MPCDI 2D Media profile.

The 4Kx42 models also support MPCDI, but at Level 1 and Level 2 only. We recommend Series 4 or Series 3 models for optimal rendering speeds.

MEDIA TYPES

MPCDI is supported with graphics objects such as *roImageWidget*, *roHtmlWidget*, and *roTextWidget*. It does not support the *roVideoPlayer* object ; to apply MPCDI to a video, display the video as part of an HTML page.

BrightScript API for MPCDI

The BrightScript API is designed such that information corresponding to a particular region must be extracted from the *.mpcdi* file and then passed to the *roVideoMode.SetMpcdi()* method. Once the configuration information for a region is passed into the player, MPCDI is enabled, and media displayed in HTML, image, and text widgets will be warped according to the configuration.

Since the MPCDI configuration file might contain more than one region, the autorun script on the player will need to select a particular region from the *.mpcdi* file and parse the data for that region before using it to configure the player. Note that each player can display one MPCDI region only.

If you wish to bypass writing a script to parse MPCDI data and configure the player, you can use the template script attached to this page (see the **MPCDI Template Script** section below for more details).

ENABLING MPCDI

To enable MPCDI, call the *roVideoMode.SetMpcdi()* method:

SetMpcdi(parameters As roAssociativeArray) As Boolean

Enables MPCDI using an associatve array of parameters. This method returns `true` if MPCDI has been enabled and `false` if it could not be enabled. The associative array must contain the following parameters:

- `[roAssociativeArray] region`: An associative array of parameters containing region data:

    - `[int] xresolution`: The viewport width, which corresponds to the `<XResolution>` attribute in the `<region>` tag of the *mpcdi.xml* configuration file
    - `[int] yresolution`: The viewport height, which corresponds to the `<YResolution>` attribute in the `<region>` tag of the *mpcdi.xml configuration* file
    - `[float] x`: The region coordinate horizontal position, which corresponds to the `<x>` attribute in the `<region>` tag of the *mpcdi.xml configuration* file
    - `[float] y`: The region coordinate vertical position, which corresponds to the `<y>` attribute in the `<region>` tag of the *mpcdi.xml configuration* file
    - `[float] xsize`: The region coordinate width, which corresponds to the `<xsize>` attribute in the `<region>` tag of the *mpcdi.xml configuration* file
    - `[float] ysize`: The region coordinate hieght, which corresponds to the `<ysize>` attribute in the `<region>` tag of the *mpcdi.xml configuration* file
- `[roArray] blendmaps`: An array containing one or two entries describing blend map data. The array must contain an entry describing alpha map parameters, and may contain a second entry describing beta map parameters. Each entry must have the following parameters:
    - `[string] mapname`: The map name, which can be either "alpha" or "beta"
    - `[int] width`: The map width, as provided in the map *.png* file
    - `[int] height`: The map height, as provided in the map *.png* file
    - `[int] comdepth`: The number of components held in the map, which corresponds to the `<alphaMap>` or `<betaMap>` tag in the *mpcdi.xml configuration* file
    - `[int] size`: The size of the decoded *.png* file data (in bytes)
    - `[float] gammacorrection`: The gamma correction value, which corresponds to the `<gammaEmbedded>` attribute in the `<alphaMap>` tag of the *mpcid.xml* configuration file (this setting does not apply to a beta map)
    - `[roByteArray] data`: The decoded *.png* data
- `[roAssociativeArray] warp`: An associative array of parameters containing warp data:
    - `[int] width`: The width of the geometry warp map, which is provided in the *.pfm* file
    - `[int] height`: The height of the geometry warp map, which is provied in the *.pfm* file
    - `[roArray] data`: The extracted *.pfm* file data as an array of float values

DISABLING MPCDI

To disable MPCDI, pass Invalid to the `SetMpcdi()` method.

Example

```
vm = CreateObject("roVideoMode")
mpcdi_params = invalid
vm.SetMpcdi(mpcdi_params)
```

MPCDI Template Script

The *.brs* script attached to this page will extract all relevant data from an *.mpcdi* file and use it to configure MPCDI on the player. It then displays up to three HTML widgets: One widget can be used to display a video or image as a simple HTML page, while the other two display HTML pages from the web.

The `Main()` function accepts an array of parameters:

```
[mpcdi_enabled, widget_count, filename.mpcdi, mode, ip_media_filename]
```

- `[string] mpcdi_enabled`: A flag that determines whether MPCDI is enabled ("on") or disabled ("off")
- `[int] widget_count`: The number of widgets to display on screen (1, 2, or 3). If you specify a single widget, the image/video widget will be displayed as full screen.
- `[string] filename.mpcdi`: The name of the *.mpcdi* file
- `[string] mode`: The mode of the image/video widget ("image" or "video")
- `[string] ip_media_filename`: The name of the media file to display in the image/video widget.

To modify the HTML pages displayed by the second and third widgets, change the `SetUrl()` strings in the `ShowUserWidgets()` function.

Setting MPCDI Regions

If you are using the template script attached to this page, you will need to assign each player to an MPCDI region before running the script. This is done by writing the region assignment to the player registry. The template script will retrieve the value from the registry and use it to extract the correct region data from the *.mpcdi* file.

You can write region assignments to the registry using the BrightSign Shell, or you can use BrightScript.

BRIGHTSIGN SHELL

Perform the following steps on each BrightSign player that is part of the MPCDI display:

1. Connect the player to a PC using a serial cable or Telnet/SSH.
2. Power off the player and remove the microSD card or other storage.
3. Power on the player.
4. Wait until the BrightSign splash screen appears on the display.
5. Depress the SVC button on the player. This will result in entry into the `BrightSign>` shell.
6. Type "registry write brightscript mpcdi_region [region id]", where [region id] refers to the id attribute in the *mpcdi.xml* file associated with the region you wish to assign the player. Press Enter when finished.
7. Type "reboot" and press Enter.

If you later need to retrieve the region ID stored in the player registry, you can enter the following command into the shell: "registry read brightscript mpcdi_region"

BRIGHTSCRIPT

Include the following in your autorun, plugin, or standalone script:

```
reg = CreateObject("roRegistrySection", "brightscript")
reg.write("mpcdi_region","region id")
```

The "region id" refers to the `id` attribute in the *mpcdi.xml* file associated with the region you wish to assign the player. If this code is part of an autorun, you will need to develop a mechanism to differentiate among players and assign the correct region ID automatically.

You will also need to reboot the player at some point (for example, by calling `RebootSystem()`) before displaying MPCDI so that the registry write goes into effect.

If you later need to retrieve the region ID stored in the player registry, you can call the *roRegistrySection.Read()* method.