DISEScript

DISEScript api documentation

Audio

Audio | DISEScript.GetMute(:layer?)

Queries the mute state of a layer and returns the value as a boolean value. Only works in content using the new CX Portal Layer parameter is optional, if left out Mute will be returned for the Display

GET

```
/GetMute/:layer
```

- JScript [#examples-Audio-GetMute-0_0_0-0]

```
var result = DISEScript.GetMute("Layer01")
var displayResult = DISEScript.GetMute()
```

Parameter

Field	Type	Description

Field	Type	Description
`layer` optional	`String`	Name of layer

Success 200

Field	Type	Description
`result`	`Boolean`	Mute state of layer

Audio | DISEScript.GetVolume(:layer?)

Queries the mute state of a layer and returns the value as a boolean value. Only works in content using the new CX Portal Layer parameter is optional, if left out Mute will be returned for the Display

GET

```
/GetVolume/:layer
```

- JScript [#examples-Audio-GetVolume-0_0_0-0]

```
var volume = DISEScript.GetVolume("Layer01")
var displayVolume = DISEScript.GetVolume()
```

Parameter

Field	Type	Description
`layer` optional	`String`	Name of layer

Success 200

Field	Type	Description
`volume`	`Number`	Audio level of layer (0-100)

Audio | DISEScript.SetMute(:layer?, :muteState)

Sets the mute state of a layer. Only works in content using the new CX Portal Layer parameter is optional, if left out Mute will be set on the Display

GET

```
/SetMute/:layer/:muteState
```

- JScript [#examples-Audio-SetMute-0_0_0-0]

```
DISEScript.SetMute("Layer01", true)
DISEScript.SetMute(true)
```

Parameter

Field	Type	Description
`layer` `optional`	`String`	Name of layer
`muteState`	`Boolean`	Mute state of the layer

Audio | DISEScript.SetVolume(:layer?, :volume)

Sets the volume level of a layer. Only works in content using the new CX Portal Layer parameter is optional, if left out Volume will be set on the Display

GET

```
/SetVolume/:layer/:volume
```

- JScript [#examples-Audio-SetVolume-0_0_0-0]

```
DISEScript.SetVolume("Layer01", 50)
DISEScript.SetVolume(50)
```

Parameter

Field	Type	Description
`layer` `optional`	`String`	Name of layer
`volume`	`Number`	Audio level of layer (0-100)

Datafeed

Datafeed | DISEScript.GetDatafeed(:name, :mediaType?)

DEPRECATED

Queries the portal for a datafeed with the given name and returns its value (classic portal template) If mediatype is specified, will only search for datafeeds of that mediatype. Can only be used from Load scrips in classic portal templates

GET

```
/GetDatafeed/:name/:mediaType?
```

- Media types [#examples-Datafeed-GetDatafeed-0_0_0-0]
- JScript [#examples-Datafeed-GetDatafeed-0_0_0-1]

```
text
text/plain
image
image/*
video
video/*
TODO: Add more media types...
```

```
var text = DISEScript.GetDatafeed("Text")
var imageFile = DISEScript.GetDatafeed("Image")
```

Parameter

Field	Type	Description
`name`	`String`	Datafeed field name
`mediaType` `optional`	`String`	Media type

Success 200

Field	Type	Description
`result`	`String`	Contents of datafeed (empty string if datafeed was not found)

Events

Events | DISEScript.AddEvent(:name, :param)

Add event to internal queue Intended for testing or internal signaling.

GET

```
/AddEvent/:name/:param
```

- JScript [#examples-Events-AddEvent-0_0_0-0]

```
DISEScript.AddEvent("Test", "Parameter")
```

Parameter

Field	Type	Description
`name`	`String`	Name of event
`param`	`String`	Param (can be empty string)

Events | DISEScript.ProcessEvents(:sleepTime?)

Process events (Message pump). If there are events in the queue calling this function will send them to the callback server. Will only send events that where registered using RegisterEventCallback.

GET

```
/ProcessEvents/:sleepTime?
```

- Status values [#examples-Events-ProcessEvents-0_0_0-0]
- JScript [#examples-Events-ProcessEvents-0_0_0-1]
- Python [#examples-Events-ProcessEvents-0_0_0-2]

```
Terminated = Script is being terminated and you should exit as soon as possible.
Events = There are more events in the queue.
Idle = Script is idle, not terminated and there are no events in the queue at the moment.
```

```
// COM server with exposed function "Callback"
var callbackServer = {
    Callback: function(name, param) {
      DISEScript.Log('Received event: ' + name + ' with param ' + param)
    }
};

DISEScript.RegisterEventCallback('Test', callbackServer)

while (DISEScript.ProcessEvents(10) != 'Terminated') {}

DISEScript.UnregisterEventCallback('Test', callbackServer)
```

```
import pythoncom
import win32com.server.util
import win32com.server.connect

# COM server with exposed function "Callback"
class ConnectableServer(win32com.server.connect.ConnectableServer):
    _public_methods_ = ["Callback"] + win32com.server.connect.ConnectableServer._public_methods_
    def Callback(self, name, param):
        DISEScript.Log('Received event: ' + name + ' with param ' + param)

# Make an COM Dispatch object out of the "server"
callbackServer = win32com.client.dynamic.Dispatch(win32com.server.util.wrap(ConnectableServer()))

DISEScript.RegisterEventCallback('Test', callbackServer)
```

```
while DISEScript.ProcessEvents(10) != 'Terminated':
    pass

DISEScript.UnregisterEventCallback('Test', callbackServer)
```

Parameter

Field	Type	Description
`sleepTime` optional	`Number`	Time to sleep after processing events.

Success 200

Field	Type	Description
`status`	`String`	Status of the script (Terminated, Events or Idle)

Events | DISEScript.RegisterEventCallback(:name, :dispatch)

Register event callback

GET

```
RegisterEventCallback/:name/:dispatch
```

- Event Types [#examples-Events-RegisterEventCallback-0_0_0-0]
- JScript [#examples-Events-RegisterEventCallback-0_0_0-1]
- Python [#examples-Events-RegisterEventCallback-0_0_0-2]

```
Supported callback names
    Test          // Param: Optional string
    SetVolume     // Param: n/a
    TriggerOn     // Param: Name of trigger
    TriggerOff    // Param: Name of trigger
    SceneBegin    // Param: Name of layer;channel;scenario;scene
    SceneEnd      // Param: Name of layer;channel;scenario;scene
    ScenarioBegin // Param: Name of layer;channel;scenario
    ScenarioEnd   // Param: Name of layer;channel;scenario
```

```
// COM server with exposed function "Callback"
var callbackServer = {
    Callback: function(name, param) {
        DISEScript.Log('Received event: ' + name + ' with param ' + param)
    }
};

DISEScript.RegisterEventCallback('Test', callbackServer)
```

```
import pythoncom
import win32com.server.util
import win32com.server.connect

# COM server with exposed function "Callback"
```

```python
class ConnectableServer(win32com.server.connect.ConnectableServer):
    _public_methods_ = ["Callback"] + win32com.server.connect.ConnectableServer._public_methods_
    def Callback(self, name, param):
        DISEScript.Log('Received event: ' + name + ' with param ' + param)

# Make an COM Dispatch object out of the "server"
callbackServer = win32com.client.dynamic.Dispatch(win32com.server.util.wrap(ConnectableServer()))

DISEScript.RegisterEventCallback('Test', callbackServer)
```

Parameter

Field	Type	Description
`name`	`String`	Name of event
`dispatch`	`Dispatch`	Callback server

Events | DISEScript.UnregisterEventCallback(:name, :dispatch)

Unregister event callback

GET

```
/UnregisterEventCallback/:name/:dispatch
```

- JScript [#examples-Events-UnregisterEventCallback-0_0_0-0]

```
DISEScript.UnregisterEventCallback('Test', callbackServer)
```

Parameter

Field	Type	Description
`name`	`String`	Name of event
`dispatch`	`Dispatch`	Callback server

General

General | DISEScript.Action(:name, :parameter1, :parameter2)

Custom actions

GET

```
/Action/:name/:parameter1/:parameter2
```

- Available actions [#examples-General-Action-0_0_0-0]
- JScript [#examples-General-Action-0_0_0-1]

```
UpdateFile
  Triggers a file update.
  parameter1 is the destination file, parameter2 is the source file.
  If source is omitted the destination file is reloaded.
  Source file will be removed if specified.
TakeOver
  Creates a new layer for a specific media, playing once.
  All other material is paused during this playback.
  parameter1 is the file path, parameter2 is the media type.
```

```
DISEScript.Action("TakeOver", "video.mp4", "video/mp4")
```

Parameter

Field	Type	Description
`name`	`String`	Action name
`parameter1`	`String`	Parameter1

Field	Type	Description
`parameter2`	`String`	Parameter2

Success 200

Field	Type	Description
`result`	`String`	Result of action

General | DISEScript.Exit()

Immediately exits the currently running script (will continue with the next script if multiple scripts are defined)

GET

```
/Exit
```

- JScript [#examples-General-Exit-0_0_0-0]

```
DISEScript.Exit()
```

General | DISEScript.GetFilePath(:filename)

Get full pathname for filename Will look for an file with the specified name in content downloaded by player and return the complete filename. The filename is not case sensitive but should contain the file extention.

GET

```
/GetFilePath/:filename
```

- JScript [#examples-General-GetFilePath-0_0_0-0]

```
DISEScript.GetFilePath("image.jpg")
```

Parameter

Field	Type	Description
`filename`	`String`	Filename to look for

Success 200

Field	Type	Description
`filename`	`String`	Full path for filename. ex. "C:\DiseContent\01\FE\12345678912334567ABCDE"

General | DISEScript.GetState(reserved)

DEPRECATED Use ProcessEvents function instead.

If the script is running, will return Normal, if it is not running, Terminated.

GET

```
/GetState/:reserved
```

- Status values [#examples-General-GetState-0_0_0-0]
- JScript [#examples-General-GetState-0_0_0-1]

```
Terminated = Script is being terminated and you should exit as soon as possible.
Normal = Script is idle
```

```
var state = DISEScript.GetState(0)
```

Parameter

Field	Type	Description
`reserved`	`Number`	Reserved value, should be "0"

General | DISEScript.Log(:level?, :message)

Log a message to DISE portal.

GET

```
/Log/:level?/:message
```

- Log levels [#examples-General-Log-0_0_0-0]
- JScript [#examples-General-Log-0_0_0-1]

```
If loglevel is specified, it needs to be one of:
debug
info
notice
warning
error
critical
alert
emergency
```

```
DISEScript.Log('Help')
DISEScript.Log('error', 'Help')
```

Parameter

Field	Type	Description
`level` optional	`String`	Log level (default value is debug)
`message`	`String`	Message

General | DISEScript.Sleep(:milliseconds)

Delays the script execution by x milliseconds.

GET

```
/Sleep/:milliseconds
```

- JScript [#examples-General-Sleep-0_0_0-0]

```
DISEScript.Sleep(1000) // Sleep for one second
```

Parameter

Field	Type	Description

Field	Type	Description
`milliseconds`	`Number`	Time to sleep in milliseconds

Playback

Playback | DISEScript.Back(:name?)

Sends a back command to the currently playing playlist with the provided name on the player.

GET

```
/Back/:name?
```

- JScript [#examples-Playback-Back-0_0_0-0]

```
DISEScript.Back()
DISEScript.Back("ProductFocus")
```

Parameter

Field	Type	Description

Field	Type	Description
name `optional`	`String`	Scenario name. If not supplied current scenario will forward

---

Playback | DISEScript.Forward(:name?)

Sends a forward command to the currently playing playlist with the provided name on the player.

GET

```
/Forward/:name?
```

- JScript [#examples-Playback-Forward-0_0_0-0]

```
DISEScript.Forward()
DISEScript.Forward("ProductFocus")
```

Parameter

Field	Type	Description
name `optional`	`String`	Scenario name. If not supplied current scenario will forward

Trigger

Trigger | DISEScript.GetTrigger(:name)

Queries the playback engine for a trigger with the given name and returns its value as a boolean value.

GET

```
/GetTrigger/:name
```

- JScript [#examples-Trigger-GetTrigger-0_0_0-0]

```
var triggerState = DISEScript.GetTrigger("Trigger01")
```

Parameter

Field	Type	Description
`name`	`String`	Trigger name

Success 200

Field	Type	Description

Field	Type	Description
`result`	`Boolean`	Current state of the trigger

Trigger | DISEScript.SetTrigger(:name, :value)

Sets a trigger in the portal with the name provided.

GET

```
/SetTrigger/:name/:value
```

- JScript [#examples-Trigger-SetTrigger-0_0_0-0]

```
DISEScript.SetTrigger("Trigger01", true)
```

Parameter

Field	Type	Description
`name`	`String`	Trigger name

Field	Type	Description
`value`	`Boolean`	New value of trigger

Trigger | DISEScript.ToggleTrigger(:name)

Sets the trigger. To true if it is already false, to false if it is true.

GET

```
/ToggleTrigger/:name
```

- JScript [#examples-Trigger-ToggleTrigger-0_0_0-0]

```
DISEScript.ToggleTrigger("Trigger01")
```

Parameter

Field	Type	Description
`name`	`String`	Trigger name

Variable

Variable | DISEScript.GetVariable(:scope?, :name)

Queries the portal for a variable with the given name and returns its value. If the variable is nonexistant, an empty string is returned. If 2 parameters are provided, the first parameter is interpreted as the scope of the variable.

GET

```
/GetVariable/:scope?/:name
```

- Scopes [#examples-Variable-GetVariable-0_0_0-0]
- JScript [#examples-Variable-GetVariable-0_0_0-1]

```
global
  The default for background scripts.
local
  The default for load scripts. Variable will only be visible to the containing movie/template.
```

```
var text = DISEScript.GetVariable("Text")
var imageFile = DISEScript.GetVariable("Image")
```

Parameter

Field	Type	Description
`scope` optional	`String`	Scope (Local, Global)
`name`	`String`	Variable name

Success 200

Field	Type	Description
`result`	`String`	Contents of variable (empty string if variable was not found)

Variable | DISEScript.SetVariable(:scope?, :name, :value)

Sets a variable in the portal. If 3 parameters are provided, the first parameter is interpreted as the scope of the variable.

GET

```
/SetVariable/:scope?/:name/:value
```

- Scopes [#examples-Variable-SetVariable-0_0_0-0]
- JScript [#examples-Variable-SetVariable-0_0_0-1]

```
global
  The default for background scripts.
local
  The default for load scripts. Variable will only be visible to the containing movie/template.
```

```
DISEScript.SetVariable("Variable1", "some value)
DISEScript.SetVariable("Local", "Variable1", "some value)
DISEScript.SetVariable("Global", "GlobalVariable1", "another value")
```

Parameter

Field	Type	Description
`scope` `optional`	`String`	Scope (Local, Global)
`name`	`String`	Name
`value`	`String`	Value

Generated with apidoc [https://apidocjs.com] 0.51.1 - Fri Aug 12 2022 16:41:57 GMT+0200 (Central European Summer Time)