Premium HTTP Api

Documentation for the HTTP Api access provided by the Dise Premium and Professional players

Dise Premium comes with an built-in web server. The server provides information and control by a standard REST format. To access a player using the HTTP API do the following:

- First make sure the player has enabled the web server [https://support.dise.com/a/solutions/articles/9000081899#Network] .
- Open a web browser and enter "http://[player IP or hostname]:8080". A web page should appear. If you get a 404 error, please make sure the player has a network connection [https://support.dise.com/en/support/solutions/articles/9000095194-network-access-security] .

In the HTTP API, requests that you send to the player either by GET or some other HTTP method. GET requests are generally information about the state of the player, whereas POST or PUT lets you control the playback. When you make a POST or PUT command, make sure the "Content-Type" header is set to "text/xml" ("Content-Type: text/xml")

For the examples given below, we are using curl [http://curl.haxx.se/]

API

API | Methods

Retrieve list of API methods Root folder for the API. Here you get a list of the API methods that are registered. These methods can be called to get further information for each by accessing url /api/ Only Replay method is supported at the moment

GET

```
/api
```

- Response [#success-examples-API-API_Methods-1_0_0-0]

```
HTTP/1.1 200 OK
<?xml version="1.0" encoding="UTF-8" ?>
<methods>
    <method>ComputerStatus</method>
    <method>ContentTarget</method>
    <method>Probe</method>
    <method>Replay</method>
    <method>ScreenControl</method>
    <method>StatisticsMonitor</method>
</methods>
```

HTML

HTML | Info page

Get HTML page with replay info

GET

```
/replay_info.html
```

- Response [#success-examples-HTML-Replay_info-1_0_0-0]

```
HTTP/1.1 200 OK
```

HTML | Start page

Get HTML page with start page

GET

```
/index.html
```

- Response [#success-examples-HTML-Start-1_0_0-0]

```
HTTP/1.1 200 OK
```

Replay

Replay | Get log

Get the latest log information.

GET

```
/api/Replay/:id/Log
```

Parameter

Field	Type	Description
id	String	• Id

- Response [#success-examples-Replay-Replay_log-1_0_0-0]

```
HTTP/1.1 200 OK
<Replay>
  <Log>
    <![CDATA[220627-142832: Playback(400) - Debug(7): Scene::Start(): Scene 1 594626,23
220627-142842: Playback(400) - Debug(7): Scene::Stop(): Scene 1
220627-142842: Playback(400) - Debug(7): Scene::Start(): Scene 2 594636,23
220627-142852: Playback(400) - Debug(7): Scene::Stop(): Scene 2
220627-142852: Playback(400) - Debug(7): Scene::Start(): Scene 1 594646,23]]>
  </Log>
</Replay>
```

Replay | Get performance monitor info

Get performance monitor output

GET

```
/api/Replay/:id/PerformanceMonitor
```

Parameter

Field	Type	Description
id	String	• Id

- Response [#success-examples-Replay-Replay_performance_monitor-1_0_0-0]

```
HTTP/1.1 200 OK
<Replay>
  <PerformanceMonitor>
    <Counter Name="Display scheme Render" ReportCount="47480" AvgTime="0,0001066" AvgCounter="1066" LatestTime="0,
    <Counter Name="Render a Frame" ReportCount="47480" AvgTime="0,0003435" AvgCounter="3435" LatestTime="0,0002729
  </PerformanceMonitor>
</Replay>
```

Replay | Get screenshot

Get the current playing screenshot

GET

```
/api/Replay/:id/Screenshot?width=:width&height=:height
```

- Example: Get a 200 pixels wide thumbnail screenshot [#examples-Replay-Screenshot-1_0_0-0]

```
curl -X GET http://localhost:8080/api/Replay/%7BDC61460D-37BD-420F-B859-F21E25B2BC82%7D/Screenshot?width=200
```

Parameter

Field	Type	Description
`id`	`String`	• Id
`width` `optional`	`Number`	• The width of the screenshot, in pixels. When this is set, the height will be set automatically to keep the aspect ratio.
`height` `optional`	`Number`	• The height of the screenshot, in pixels.

- Response [#success-examples-Replay-Screenshot-1_0_0-0]

```
HTTP/1.1 200 OK
```

Replay | **Instance**

Basic information (settings, display schemes, etc) When we refer to ":id", it is the software ID that needs to be specified. An example is: {DC61460D-37BD-420F-B859-F21E25B2BC82} When making a HTTP request, the { and } need to be replaced with %7B and %7D, respectively, making it: %7BDC61460D-37BD-420F-B859-F21E25B2BC82%7D

GET

```
/api/Replay/:id
```

Parameter

Field	Type	Description
`id`	`String`	• Software id

Success 200

Field	Type	Description
`name`	`String`	Software id for this replay
`FriendlyName`	`String`	Name of replay instance

Field	Type	Description
`DisplayScheme`	`String`	Filename of the selected displayscheme / content file that is playing

- Response [#success-examples-Replay-Replay_instance-1_0_0-0]

```
HTTP/1.1 200 OK
<Replay>
  <Instance name="{CD740CF6-90C4-4284-BD60-33757AD0AF05}">
    <FriendlyName>KSDCPU01</FriendlyName>
    <DisplayScheme>C:\DiseContent\Content\test-content.disemovie</DisplayScheme>
  </Instance>
</Replay>
```

Replay | List instances

Retrieve list of Replays and their software IDs. Communication with Dise Replay (Playback enginge). Several instances of Replay on the same player can be controlled by the API - we therefore need to indicate to which instance of Replay to send. This page gives us a list of the Replays on the player and their software ID. Use this ID for all subsequent requests to Replay.

GET

```
/api/Replay
```

Success 200

Field	Type	Description
name	String	Software id for this replay
FriendlyName	String	Name of replay instance

- Response [#success-examples-Replay-Replay_list-1_0_0-0]

```
HTTP/1.1 200 OK
<Replay>
  <Instances>
    <Instance name="{CD740CF6-90C4-4284-BD60-33757AD0AF05}">
      <FriendlyName>KSDCPU01</FriendlyName>
    </Instance>
  </Instances>
</Replay>
```

Replay | Set display scheme

Set the display scheme of the playback

PUT

```
/api/Replay/:id/DisplayScheme
```

- Example: Send command to pause [#examples-Replay-Replay_set_displayscheme-1_0_0-0]

```
curl -H "Content-Type:text/xml" -X PUT --data "<Request><Playback><Layer><Command>Volume</Command><Value>50</Value
```

Parameter

Field	Type	Description
`id`	`String`	• Id
`DisplayScheme`	`String`	• Filename to display scheme

- Request [#parameter-examples-Replay-Replay_set_displayscheme-1_0_0-0]

```
<Request>
    <DisplayScheme>C:\Temp\Display.displayscheme</DisplayScheme>
</Request>
```

- Response [#success-examples-Replay-Replay_set_displayscheme-1_0_0-0]

```
HTTP/1.1 202 Accepted
```

Replay OSD

Replay OSD | Get OSD settings

Get On Screen Display settings.

GET

```
/api/Replay/:id/OSD
```

Parameter

Field	Type	Description
id	String	• Id

- Response [#success-examples-Replay_OSD-Get_OSD_settings-1_0_0-0]

```
HTTP/1.1 200 OK
<Replay>
  <OSD>
    <OSDLevel>0</OSDLevel>
    <OSDColor>1</OSDColor>
  </OSD>
</Replay>
```

Replay OSD | Set OSD settings

Set On Screen Display settings.

PUT

```
/api/Replay/:id/OSD
```

- Example: Show the OSD page with playback information [#examples-Replay_OSD-Set_OSD_settings-1_0_0-0]

```
curl -H "Content-Type:text/xml" -X PUT --data "<Request><OSD><OSDLevel>2</OSDLevel></OSD></Request>" http://localh
```

Parameter

Field	Type	Description
`id`	`String`	• Id
`OSDLevel`	`Request.OSD.OSDLevel`	• Level (0 = off, 1 = FPS counter, 2 = full OSD)
`OSDColor`	`Request.OSD.OSDColor`	• Color (0 = black, 1 = white, ...)

- Request [#parameter-examples-Replay_OSD-Set_OSD_settings-1_0_0-0]

```
<Request>
    <OSD>
        <OSDLevel>2</OSDLevel>
        <OSDColor>1</OSDColor>
    </OSD>
</Request>
```

- Response [#success-examples-Replay_OSD-Set_OSD_settings-1_0_0-0]

```
HTTP/1.1 202 Accepted
```

Replay playback

Replay playback | Get channel

Get channel by name Supported with Professional Replay and Premium using viewer = Replay

GET

```
/api/Replay/:id/Playback/Channel/:name
```

Parameter

Field	Type	Description
`id`	`String`	• Replay Id
`name`	`String`	• Channel name

Success 200

Field	Type	Description
`Name`	`String`	Name of channel. Will use name of dise movie if no channel is specified.
`State`	`String`	State of channel. Stopped, Running or Paused

- Response [#success-examples-Replay_playback-Replay_channel-1_0_0-0]

```
HTTP/1.1 200 OK
<Replay>
  <Playback>
    <Channel>
      <Name>Main</Name>
      <State>Running</State>
    </Channel>
  </Playback>
</Replay>
```

Replay playback | Get layer

Get layer Supported with default Premium setup

GET

```
/api/Replay/:id/Playback/Layer/:name
```

Parameter

Field	Type	Description
`id`	`String`	• Replay Id
`name`	`String`	• Layer name

Success 200

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

Field	Type	Description
`Volume`	`Number`	Volume of playback in percent (0-100)
`Mute`	`Boolean`	Mute as boolean (true/false)

- Response [#success-examples-Replay_playback-Replay_layer-38_0_0-0]

```
HTTP/1.1 200 OK
<Replay>
  <Playback>
    <Layer>
      <Name>Layer 1</Name>
      <Volume>100</Volume>
      <Mute>false</Mute>
    </Layer>
  </Playback>
</Replay>
```

Replay playback | Get playback info

Get playback status (stopped, playing, pause, display scheme, forward, backward)

GET

```
/api/Replay/:id/Playback
```

Parameter

Field	Type	Description
`id`	`String`	• Software id

Success 200

Field	Type	Description
`Status`	`String`	Info about channels, movie and scenes currently playing
`FramesPerSecond`	`Number`	FPS
`State`	`String`	State of playback (stopped, playing)
`DisplayScheme`	`String`	Filename of the selected displayscheme / content file that is playing

Field	Type	Description
`CurrentPlayingContentName`	`String`	The currently playing content
`UpdateCount`	`Number`	Number of files left to update, zero if no update is going on.
`Volume` `optional`	`Number`	Volume of playback in percent (0-100). Optional, will only show up when playing CX Portal content.
`Mute` `optional`	`Boolean`	Mute as boolean (true/false). Optional, will only show up when playing CX Portal content.

- Response stopped [#success-examples-Replay_playback-Replay_playback-38_0_0-0]
- Response playing [#success-examples-Replay_playback-Replay_playback-38_0_0-1]
- Response CX Portal [#success-examples-Replay_playback-Replay_playback-38_0_0-2]

```
HTTP/1.1 200 OK
<Replay>
  <Playback>
    <Status/>
    <FramesPerSecond/>
    <State>stopped</State>
    <DisplayScheme>C:\DiseContent\Content\test\test-content.disemovie</DisplayScheme>
    <CurrentPlayingContentName/>
    <UpdateCount>0</UpdateCount>
```

```
  </Playback>
</Replay>
```

```
HTTP/1.1 200 OK
 <Replay>
  <Playback>
    <Status>
      <Channel>
        <Name>Main</Name>
        <State>Running</State>
        <Movie>
          <Name>test-content.disemovie</Name>
          <Scene>
            <Name/>
            <State>Running</State>
            <Progress>35,37</Progress>
            <Duration>10</Duration>
            <LastStartTime>2022-06-27 14:19:12</LastStartTime>
          </Scene>
          <Scene>
            <Name/>
            <State>Loaded</State>
            <Progress>0</Progress>
            <Duration>10</Duration>
            <LastStartTime/>
          </Scene>
        </Movie>
      </Channel>
    </Status>
```

```
    <FramesPerSecond>58,9379440454754</FramesPerSecond>
    <State>playing</State>
    <DisplayScheme>C:\DiseContent\Content\test\test-content.disemovie</DisplayScheme>
    <CurrentPlayingContentName>test-content.disemovie</CurrentPlayingContentName>
    <UpdateCount>0</UpdateCount>
  </Playback>
</Replay>
```

```
HTTP/1.1 200 OK
<Replay>
  <Playback>
    <Status>
      <Channel>
        <Name>Default</Name>
        <State>Running</State>
      </Channel>
    </Status>
    <FramesPerSecond>59,9670790566648</FramesPerSecond>
    <State>playing</State>
    <DisplayScheme>C:\Users\User1\AppData\Local\Temp\DISE\DISE 8\Display_46672.displayscheme</DisplayScheme>
    <CurrentPlayingContentName>Test1:test-content</CurrentPlayingContentName>
    <UpdateCount>0</UpdateCount>
    <Volume>100</Volume>
    <Mute>false</Mute>
  </Playback>
</Replay>
```

Replay playback | Get playback info

Get playback status (stopped, playing, pause, display scheme, forward, backward)

GET

```
/api/Replay/:id/Playback
```

Parameter

Field	Type	Description
`id`	`String`	• Software id

Success 200

Field	Type	Description
`Status`	`String`	Info about channels, movie and scenes currently playing
`FramesPerSecond`	`Number`	FPS
`State`	`String`	State of playback (stopped, playing)

Field	Type	Description
`DisplayScheme`	`String`	Filename of the selected displayscheme / content file that is playing
`CurrentPlayingContentName`	`String`	The currently playing content
`UpdateCount`	`Number`	Number of files left to update, zero if no update is going on.

- Response stopped [#success-examples-Replay_playback-Replay_playback-1_0_0-0]
- Response playing [#success-examples-Replay_playback-Replay_playback-1_0_0-1]

```
HTTP/1.1 200 OK
<Replay>
  <Playback>
    <Status/>
    <FramesPerSecond/>
    <State>stopped</State>
    <DisplayScheme>C:\DiseContent\Content\test\test-content.disemovie</DisplayScheme>
    <CurrentPlayingContentName/>
    <UpdateCount>0</UpdateCount>
  </Playback>
</Replay>
```

```
HTTP/1.1 200 OK
 <Replay>
  <Playback>
    <Status>
      <Channel>
        <Name>Main</Name>
        <State>Running</State>
        <Movie>
          <Name>test-content.disemovie</Name>
          <Scene>
            <Name/>
            <State>Running</State>
            <Progress>35,37</Progress>
            <Duration>10</Duration>
            <LastStartTime>2022-06-27 14:19:12</LastStartTime>
          </Scene>
          <Scene>
            <Name/>
            <State>Loaded</State>
            <Progress>0</Progress>
            <Duration>10</Duration>
            <LastStartTime/>
          </Scene>
        </Movie>
      </Channel>
    </Status>
    <FramesPerSecond>58,9379440454754</FramesPerSecond>
    <State>playing</State>
    <DisplayScheme>C:\DiseContent\Content\test\test-content.disemovie</DisplayScheme>
```

```
    <CurrentPlayingContentName>test-content.disemovie</CurrentPlayingContentName>
    <UpdateCount>0</UpdateCount>
  </Playback>
</Replay>
```

Replay playback | List channels

List all channels in the current content Supported with Professional Replay and Premium using viewer = Replay

GET

```
/api/Replay/:id/Playback/Channel
```

Parameter

Field	Type	Description
`id`	`String`	• Replay Id

Success 200

Field	Type	Description

Field	Type	Description
`Name`	`String`	Name of channel. Will use name of dise movie if no channel is specified.
`State`	`String`	State of channel. Stopped, Running or Paused

- Response [#success-examples-Replay_playback-Replay_channels-1_0_0-0]

```
HTTP/1.1 200 OK
<Replay>
  <Playback>
    <Channel>
      <Name>Main</Name>
      <State>Running</State>
    </Channel>
  </Playback>
</Replay>
```

Replay playback | List layers

List all layers Supported with default Premium setup

GET

```
/api/Replay/:id/Playback/Layer
```

Parameter

Field	Type	Description
`id`	`String`	• Replay Id

Success 200

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

- Response [#success-examples-Replay_playback-Replay_layers-38_0_0-0]

```
HTTP/1.1 200 OK
<Replay>
  <Playback>
    <Layer>
      <Name>Layer 1</Name>
    </Layer>
  </Playback>
</Replay>
```

Replay playback | Send channel command

Send a command to the channel (play, stop, pause, unpause, restartscene, loopscene, forward, backward, gotoscene, returntoprevscene)

PUT

```
/api/Replay/:id/Playback/Channel/:name
```

- Example: Send command to pause channel [#examples-Replay_playback-Replay_channel_command-1_0_0-0]

```
curl -H "Content-Type:text/xml" -X PUT --data "<Request><Playback><Channel><Command>Pause</Command></Channel></Pla
```

Parameter

Field	Type	Description
`id`	`String`	• Id
`name`	`String`	• Name of channel (replace spaces with %20)
`Command`	`String`	• Play|Stop|Pause|Unpause|RestartScene|LoopScene|Forward|Backward|GotoScene|ReturnToPrevScene
`Param1`	`String`	• Optional parameter, needeed for GotoScene

- Request [#parameter-examples-Replay_playback-Replay_channel_command-1_0_0-0]

```
<Request>
    <Playback>
        <Channel>
            <Command>Command</Command>
            <Param1></Param1>
        </Channel>
    </Playback>
</Request>
```

- Response [#success-examples-Replay_playback-Replay_channel_command-1_0_0-0]

```
HTTP/1.1 202 Accepted
```

Replay playback | Send layer command

Send a command to the layer (volume or mute) Volume and mute will be applied to all content playing in this layer. Item volume = Windows Mixer Volume * Playback * Layer (100% * 50% * 100% = 40%) Item mute = Windows Mixer Mute * Playback * Layer (false || true || false = true)

PUT

```
/api/Replay/:id/Playback/Layer/:name
```

- Example: Send command to set volume to 50% [#examples-Replay_playback-Replay_layer_command-38_0_0-0]

```
curl -H "Content-Type:text/xml" -X PUT --data "<Request><Playback><Layer><Command>Volume</Command><Value>50</Value
```

Parameter

Field	Type	Description
`id`	`String`	• Id
`name`	`String`	• Name of layer (replace spaces with %20)
`Command`	`String`	• Volume|Mute
`Value`	`String`	• Value

- Request [#parameter-examples-Replay_playback-Replay_layer_command-38_0_0-0]

```
<Request>
    <Playback>
        <Layer>
            <Command>Volume</Command>
            <Value>50</Value>
        </Layer>
    </Playback>
</Request>
```

- Response [#success-examples-Replay_playback-Replay_layer_command-38_0_0-0]

```
HTTP/1.1 202 Accepted
```

Replay playback | Send playback command

Send a command to the playback engine (play, stop, pause, forward, backward) Volume and Mute will take an value in the request as well and are available when playing content from the CX Portal. Volume will take an value between 0 and 100 that is the volume in percent. Default is 100 Mute will take an boolean "true" or "false". Default is false Volume and mute values will apply on all content playing. Use get on Playback to see the current volume and mute levels Volumes are controlled on Video and Sound objects in the playback. Both added to playlist and inside Movie or Template.

PUT

```
/api/Replay/:id/Playback
```

- Example: Send command to pause [#examples-Replay_playback-Replay_playback_command-38_0_0-0]

```
curl -H "Content-Type:text/xml" -X PUT --data "<Request><Playback><Command>Pause</Command></Playback></Request>" h
curl -H "Content-Type:text/xml" -X PUT --data "<Request><Playback><Command>Volume</Command><Value>50</Value></Play
curl -H "Content-Type:text/xml" -X PUT --data "<Request><Playback><Command>Mute</Command><Value>true</Value></Play
```

Parameter

Field	Type	Description
`id`	`String`	• Id

Field	Type	Description
`Command`	`String`	• Play|Stop|Pause|Forward|Backward|Volume|Mute
`Value` `optional`	`String`	• Optional value for Volume and Mute

- Request [#parameter-examples-Replay_playback-Replay_playback_command-38_0_0-0]

```
<Request>
    <Playback>
        <Command>[Play|Stop|Pause|Forward|Backward|Volume|Mute]</Command>
        <Value>100</Value>
    </Playback>
</Request>
```

- Response [#success-examples-Replay_playback-Replay_playback_command-38_0_0-0]

```
HTTP/1.1 202 Accepted
```

Replay playback | Send playback command

Send a command to the playback engine (play, stop, pause, forward, backward)

PUT

```
/api/Replay/:id/Playback
```

- Example: Send command to pause [#examples-Replay_playback-Replay_playback_command-1_0_0-0]

```
curl -H "Content-Type:text/xml" -X PUT --data "<Request><Playback><Command>Pause</Command></Playback></Request>" h
```

Parameter

Field	Type	Description
`id`	`String`	• Id
`Command`	`String`	• Play|Stop|Pause|Forward|Backward|Volume|Mute

- Request [#parameter-examples-Replay_playback-Replay_playback_command-1_0_0-0]

```
<Request>
    <Playback>
        <Command>[Play|Stop|Pause|Forward|Backward]</Command>
    </Playback>
</Request>
```

- Response [#success-examples-Replay_playback-Replay_playback_command-1_0_0-0]

```
HTTP/1.1 202 Accepted
```

Replay variables

Replay variables | Delete variable

Remove variable

DELETE

```
/api/Replay/:id/Variables/:name
```

- Example: Delete variable [#examples-Replay_variables-Delete_variable-1_0_0-0]

```
curl -X DELETE http://localhost:8080/api/Replay/%7BDC61460D-37BD-420F-B859-F21E25B2BC82%7D/Variables/MyVar
```

Parameter

Field	Type	Description
`id`	`String`	• Id
`name`	`name`	• name

- Response [#success-examples-Replay_variables-Delete_variable-1_0_0-0]

```
HTTP/1.1 202 Accepted
```

Replay variables | Get variables

List current variables

GET

```
/api/Replay/:id/Variables
```

Parameter

Field	Type	Description
`id`	`String`	• Id

Success 200

Field	Type	Description
`Variable`	`String`	Name of variable
`Value`	`String`	Value of variable

Field	Type	Description
`Persistent`	`Boolean`	Persistent variables will persist between restarts
`SystemVariable`	`Boolean`	System variables are read only

- Response [#success-examples-Replay_variables-Get_variables-1_0_0-0]

```
HTTP/1.1 200 OK
<Replay>
  <Variable Value="Value" Persistent="false" SystemVariable="false">MyVar</Variable>
</Replay>
```

Replay variables | Set Variable

Set a variable with the name variable-name. The tag should equal the name parameter

PUT

```
/api/Replay/:id/Variables/:name
```

- Example: Variable management [#examples-Replay_variables-Set_variable-1_0_0-0]

```
curl -H "Content-Type:text/xml" -X PUT --data "<Request><MyVar Persistent='false'>Value</MyVar></Request>" http://
```

Parameter

Field	Type	Description
id	String	• Id
name	name	• name

- Request [#parameter-examples-Replay_variables-Set_variable-1_0_0-0]

```
<Request>
    <MyVar Persistent="false">
        Value
    </MyVar>
</Request>
```

- Response [#success-examples-Replay_variables-Set_variable-1_0_0-0]

```
HTTP/1.1 202 Accepted
```

Generated with apidoc [https://apidocjs.com] 0.51.1 - Tue Jun 28 2022 10:31:50 GMT+0200 (Central European Summer Time)