Protocol Specification

Data Display Http Control

for Digi- and VideoPosters

Version 1.0

22.06.2012

Table of Contents

1 Revision History

Date	Rev.No.	Description	Page
15.02.2012	0.1	Initial version	All
22.06.2012	1.0	Added playlistUploadDone request	

2 Introduction

2.1 Current Digi- and VideoPoster functionality for stand alone media player applications

Data Display offers several media players for digital signage applications.

- DigiPoster-III
- DigiPoster-4.3
- VideoPoster-III

These digital posters are able to receive an image show or a video show from external sources and play it on their display. The sequence of the media show is defined in a playlist file called playlist.ddx (XML format).

In this playlist.ddx file one or more media files are listed.
A sample of a playlist.ddx file is shown in the appendix of this document.
The playlist.ddx and the relevant media files must be transferred to the Data Display Poster.

Data Display provides the free Windows tool named ACC to compose playlist content and to generate a playlist.ddx file. Furthermore it also implements the file transfer to a Data Display Poster.

Additionally various configurations of a Data Display Poster can be set via the web interface.

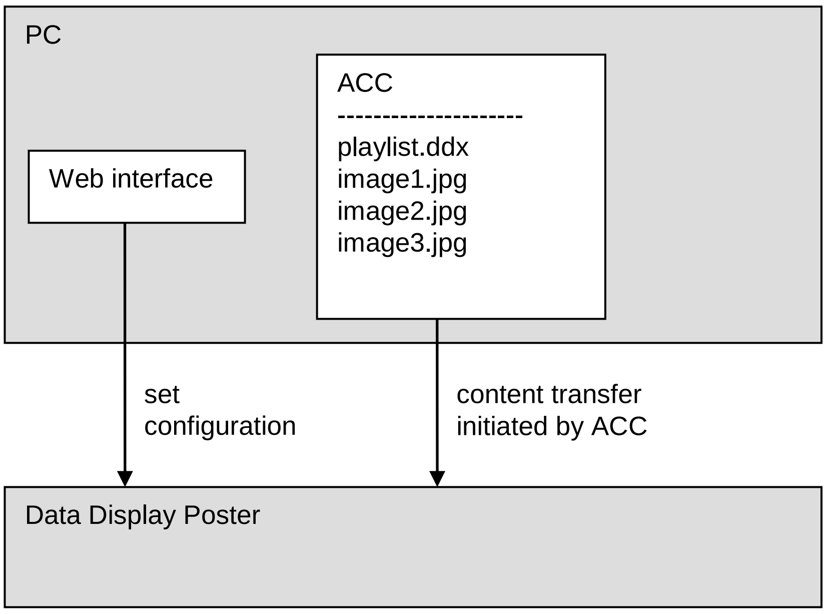


Image 1: Configuration and content update of Data Display Posters

2.2 3rd party digital signage software Integration

To integrate Digi- and VideoPosters into 3rd party digital signage software Data Display specifies a HTTP based interface. The protocol provides media content transfer, device control and configuration.

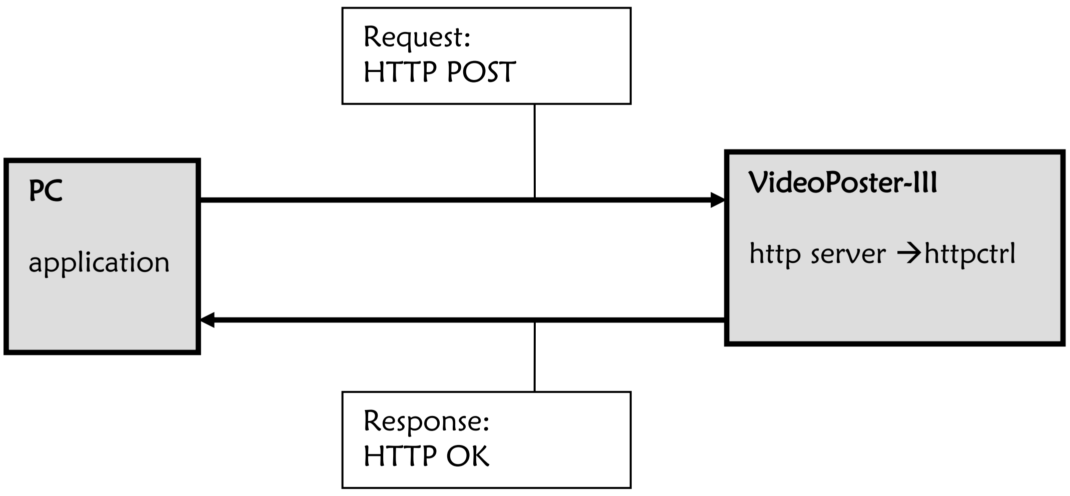
3 Http based interface to access Data Display Posters

There are 3 aspects to be realized by the interface.

1.) File transfer from a PC onto a device (e.g. playlist, playlist media, bootlogo).
    Data transfer must respect free memory-space and free disk-space.
2.) File transfer from a device to a PC, e.g. logfiles, current playlist.ddx
3.) Other device access
    a. configuration (e.g set/get brightness)
    b. control (e.g. activating triggers, display on/off)
    c. status request (get device info/versions, get media player status)
    d. request of file infos (delete file, list logfiles)

3.1 General definitions

- The **HTTP/1.1** is used as the base protocol. This means the communication is a client – server topology, while the client always initiates a request.
- The HTTP protocol defines various request methods (HEAD, GET, POST, PUT, DELETE …). For all requests defined in this protocol only the HTTP request **POST** is chosen.
- The request line and headers must all end with **<CR><LF>** (that is, a carriage return followed by a line feed).
- The HTTP header field Authorization must be transferred. The Authorization is expected in Basic 64base encoding and must be transferred with every request. Username and password are the same as set for/in the Poster web interface.



3.2 File transfer PC → Data Display Poster

File upload to the Poster device is done using Http post requests. For each file that has to be uploaded, a separate request must be made. Here is a list of requirements for a valid upload request, as well as a list of possible responses.

HTTP Request (PC to Data Display Posters):

The following table represents a HTTP request for a file transfer:

POST /httpctrl HTTP/1.1
Host: insert IP address of media player here
Authorization: Basic (insert 64base encrypted name and password)
Content-Type: **application/octet-stream**
Content- Length: insert filesize in bytes here (decimal)
Content-Disposition: **method**=”insert chosen method” **filename**=”insert filename – (how to name the file on the device)”
Content of file

HTTP headers:

Host:	the IPv4 address of the display poster to send the request to
Authorization:	The HTTP password transfer is expected to be transferred in 64base encoding.
Content-Type:	application/octet-stream
Content-Length:	Length of the file content which will be attatched to the header.
Content-Disposition:	The keyword **filename** represents the filename on the device. The keyword **method** distinguishes where to save the file on the display poster.

Methods:

Content-Disposition	
methods	**already implemented ?**
setBootlogo	No
setSetupFile	No
setMediaFile	Yes

HTTP Responses (Data Display Poster to PC):

Error Responses 4xx

Http Error Responses	**Reasons**
400 – Bad Request	- http param ‘Content-Type’ is missing/incorrect
405 – Method Not Allowed	- currently only http method POST is accepted
411 – Length Required	- header Content-Length not set
507 – Insufficient storage	- there is not enough space on the device to save the file.

Response OK:

HTTP/1.1 200 OK
Content-Length: 0
Content-Disposition: method=”repeated method of request” filename=” repeated filename of request”
Date: … (is set by the HTTP server on the device)
Server: lighttpd/1.4.28 (is set by the HTTP server on the device)

Sample

Request (PC to Data Display Poster):
POST /httpctrl HTTP/1.1
Host: 192.186.2.77
Authorization: Basic QXJXaXN0YTpXcnRpc3Rh
Content-Type: **application/octet-stream**
Content- Length: 102400
Content-Disposition: method=”setBootlogo” filename=”bootlogo.jpg”

Content of bootlogo.jpg

Response (Data Display Poster to PC):
HTTP/1.1 200 OK
Content-Length:0
Content-Disposition: method=”setBootlogo” filename=”bootlogo.jpg”
Date:Sat, 02 Jan 2010 06:23:01 GMT
Server: lighttpd/1.4.28

3.3 File transfer Data Display Poster → PC

File download from the Poster device will be implemented in some future version.

Request (Data Display Poster to PC):

The following table represents a HTTP request for a file transfer:

POST /httpctrl HTTP/1.1
Host: insert IP address of media player here
Authorization: Basic (insert 64base encrypted name and password)
Content- Length: 0
Content-Disposition: **method**=”insert chosen method” **filename**=”insert filename – (how to name the file on the device)”

HTTP headers:

Host:	the IPv4 address of the display poster to send the request to
Authorization:	The HTTP password transfer is expected to be transferred in 64base encoding.
Content-Type:	None transferred
Content-Length:	0
Content-Disposition:	The keyword **filename** represents the filename on the device. The keyword **method** distinguishes which file to get from a Data Display Poster. Those filenames can be listed/requested by requests offered in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**.

Methods:

Content-Disposition	
methods	**already implemented ?**
getBootlogo	No
getLogFile	No
getMediaFile	No
getPlaylistFile	No

HTTP Response (Data Display Poster to PC):

Error Responses 4xx

Http Error Responses	Reasons
400 – Bad Request	- http param 'Content-Type' is missing/incorrect
405 – Method Not Allowed	- currently only http method POST is accepted
404 – Not Found	- demanded file does not exist

Response OK:

HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 0
Content-Disposition: method="repeated method of request" filename=" repeated filename of request"
Date: …   (is set by the HTTP server on the device)
Server: lighttpd/1.4.28 (is set by the HTTP server on the device)
Content of requested file

Sample

Request (PC to Data Display Poster):
POST /httpctrl HTTP/1.1
Host: 192.186.2.77
Authorization: Basic QXJXaXN0YTpXcnRpc3Rh
Content- Length: 0
Content-Disposition: **method**="getMediaFile" **filename**="clip.mp4"

Response (Data Display Poster to PC):
HTTP/1.1 200 OK
Content-Type:**application/octet-stream**
Content-Disposition: **method**="getMediaFile" **filename**="clip.mp4"
Content-Length: 489000
Date:Sat, 02 Jan 2010 06:23:01 GMT
Server: lighttpd/1.4.28

Content of requested file

3.4 Xml requests

The onboard web serviceprovide xml based interaction over http protocol. In this chapter we explain how to create a valid xml request and how to interpret the xml reply received from the device.
To initiate the communication, you need to create an xml request, send the xml as a content of an http request. The device will then answer with a http reply, with an xml reply in http content.

Request (PC to Data Display Poster):

The following table lists http request settings for a valid xml request:

POST /httpctrl HTTP/1.1
Url: <host>/httpctrl
Authorization: Basic (insert 64base encrypted name and password)
Content-Type: **text/xml**
Content- Length: insert length of command structure (XML) in bytes here (decimal)
Request structure (XML), details see below

HTTP headers:

Host:	the IPv4 address of the display poster to send the request to
Authorization:	The HTTP password transfer is expected to be transferred in 64base encoding.
Content-Type:	text/xml
Content-Length:	Length of the file content which will be attatched to the header.

XML request and response definition see below

Response (Data Display Poster to PC):

Error Responses 4xx:

Http errors	**Reasons**
400 – Bad Request	- no content - http param 'Content-Length' is missing - content too long (max.: 0x1000 bytes)
405 – Method Not Allowed	- currently only http method POST is accepted

HTTP Response OK:

HTTP/1.1 200 OK
Content-Type: **text/xml**
Content-Length: Data Display Poster inserts length of reply structure (XML) in bytes here (decimal)
Date:Sat, 02 Jan 2010 06:23:01 GMT
Server: lighttpd/1.4.28
Response structure (XML), details see below

Xml Template

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="ver">
  <request id="requestId"/>
  <module name="moduleName">
    <method name="methodName">
      <arg name="argName" value="argVal"/>
    </method>
  </module>
</interface>
```

Reply

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name=”httpctrl” version="ver">
  <request id=”requestId”/>
  <return id=”returnId” message=”msg”/>
  <module name="moduleName">
    <method name="methodName">
      <arg name="argName" value="argVal"/>
    </method>
  </module>
</interface>
```

Parameters in the **request** xml structure are:
- **ver** – httpctrl protocol version the client uses.
- **requestId** – string, id to link reply to a specific request, **OPTIONAL**
- **moduleName** – string, logical module to which the request method belongs, see table on the next page
- **methodName** – one of the predefined methods, see table on the next page
- **argName**, **argVal** – definition of input arguments which are needed to execute the request (if any). For each argument one <arg> element is needed.

Parameters in the **reply** xml structure are:
- **ver** – httpctr version the device uses
- **requestId** – strin, id copied from the request id attribute and sent back
- **msg** – a human-readable reply message
- **moduleName** – module which replied, always the same as in the request
- **methodName**- method for which the reply was made, always the same as in the request
- **argName**, **argVal** – list of output arguments (if any). For each argument one <arg> element is needed.

Example

Request (PC to Data Display Poster):

```
POST /httpctrl HTTP/1.1
Host: 192.186.2.77
Authorization: Basic QXJXaXN0YTpXcnRpc3Rh
Content-Type: text/xml
Content-Length: …(in bytes)

<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="myId123"/>
  <module name="control">
    <method name="activatePlaylistTrigger">
      <arg name="triggerId" value="4"/>
    </method>
  </module>
</interface>
```

Response (Data Display Poster to PC):

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: …(in bytes)
Date:Sat, 02 Jan 2010 06:23:01 GMT
Server: lighttpd/1.4.28

<?xml version="1.0" encoding="UTF-8"?>
<interface name=”httpctrl” version="1.0.0">
  <request id=”myId123”/>
  <return id=”0”/ message=”Success”>
  <module name="control">
    <method name="activatePlaylistTrigger">
      <arg name="triggerId" value="4"/>
    </method>
  </module>
</interface>
```

Here is the list of all requests for http control, implemented and foreseen:

Methods for modules		
module	**method**	**already implemented ?**
configuration	setNetworkConfig	
	getNetworkConfig	
	setHttpPassword	
	seContentAutoUpdateConfig	
	getContentAutoUpdateConfig	
	getDisplayConfig	
	setDisplayConfig	
	getBootlogoConfig	
	setBootlogoConfig	
	setTimeZone	
	getTimeZone	
	setTimeServer	
	getTimeServer	
	setSystemTime	
	getSystemTime	
	resetFactorySettings	
control	activatePlaylistTrigger	yes
	resetNetwork	
	rebootDevice	
	playerOn	
	playerOff	
	playlistUploadDone	yes
file	listLogFiles	
	listMediaFiles	
	deleteFile	
	deleteMediaFiles	
status	getDeviceInfo	
	getPlayerStatus	yes

For each request, you must define the module to for which to send to. The modules are introduced to group similar request together.

Module names
configuration
control
file
status

The following table shows all possible return ids in the xml reply:

XML return ids	message
0 (success)	
1	Internal failure
2	Missing element
3	Missing attribute
4	Invalid interface name
5	Invalid interface version
6	Invalid request id
7	Invalid module name
8	Invalid method name
9	Invalid argument name
10	Invalid argument value
11	Invalid attribute value
12	Function failed

3.4.1 Examples

Activate playlist trigger method is used to invoke an action defined in the playlist. Here are example xml requests and replies:

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="myId01"/>
  <module name="control">
    <method name="activatePlaylistTrigger">
      <arg name="triggerId" value="4"/>
    </method>
  </module>
</interface>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name=”httpctrl” version="1.0.0">
  <request id=”myId01”/>
  <return id=”0” message=”Success”/>
  <module name="control">
    <method name="getPlayerStatus">
      <!-- no arguments! -->
    </method>
  </module>
</interface>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="foobar"/>
  <module name="control">
    <method name="xy">
      <arg name="triggerId" value="4"/>
    </method>
  </module>
</interface>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="foobar"/>
  <return id="8" message="Invalid method name: (xy)"/>
</interface>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="myId03"/>
  <module name="control">
    <method name=" activatePlaylistTrigger ">
      <arg name="triggerId" value="5"/>
    </method>
  </module>
</interface>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="myId03"/>
  <return id="12" message="Function failed: No event
contains trigger with id 5"/>
</interface>
```

Get player status is used to find out what is currently playing on the device. The second example is missing the element module and provokes an error reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="12366"/>
  <module name="status">
    <method name="getPlayerStatus"/>
  </module>
</interface>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="12366"/>
  <return id="0" message=”Success”/>
  <module name="status">
    <method name="getPlayerStatus">
      <arg name="playerState" value="Playing"/>
      <arg name="playlistName" value="Playlist 1"/>
    </method>
  </module>
</interface>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="12366"/>
  <method name="getPlayerStatus"/>
</interface>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
        <request id="12366"/>
        <return id="2" message="Missing element:
        module"/>
</interface>
```

3.5 XML module-method specification

3.5.1 Module: status

3.5.1.1 Method: getDeviceInfo

The Function getDeviceInfo returns a number of ids and version numbers which represent the device hardware and firmware.

Request
Module: status
Method: getDeviceInfo
Arguments: none

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="R1"/>
  <module name="control">
    <method name="getDeviceInfo"/>
  </module>
</interface>
```

Response
Module: status
Method: getDeviceInfo
Arguments: see the table below

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="R1"/>
  <return id="0"/>
  <module name="status">
    <method name="getDeviceInfo">
      <arg name="platformId" value="3"/>
      <arg name="productId" value="4"/>
      <arg name="productName" value="DigiPoster-4.3"/>
      <arg name="firmwareVersion" value="1.0.0.0"/>
      <arg name="kernelVersion" value="2.6.27.8"/>
      <arg name="hardwareVersion"
value="1.1_00_23208"/>
      <arg name="configurationVersion" value="1.0.0.0"/>
      <arg name="cofigurationName" value="PA-02-101"/>
    </method>
  </module>
</interface>
```

Response arguments table:

Name	Range or possible values	Comment
platformId	2	PL_AMEDIA_II - VideoPoster-III and others
	3	PL_ANET_43 – DigiPoster-4.3 and others
	4	PL_ANET_III – DigiPoster-III and others
productId	4	PR_DIPO_43 - DigiPoster-4.3
	5	PR_DIPO_III - DigiPoster-III
	6	PR_VIPO_II - VideoPoster-III
productName	DigiPoster-III	
	DigiPoster-4.3	
	VideoPoster-III	
firmwareVersion	1.0.0.0	*1)
kernelVersion	2.6.27.8	*1)
hardwareVersion	1.1_00_23208	might have different formats on the various posters
configurationVersion	1.0.0.0	*1)
cofigurationName	PA-02-101	DigiPoster-4.3
	PA-26-001	VideoPoster-III PA-26-XXX

*1) All version numbers (except hardware version) are in format: <number>.<number>.<number>.<number>

3.5.1.2 Method: getPlayerStatus

Note: The Http header details are not listed for every XML call. A detailed description of the demanded http header is to be found in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**.

Supported since firmware version:

Product name	Firmware version
DigiPoster-III	n.a
DigiPoster-4.3	n.a.
VideoPoster-III	n.a.

Description:
This function is requests the current state of the posters' media player. In case no playlist is uploaded yet, the media player state will be indicated as 'Initializing'.

Request
Module: status
Method: getPlayerStatus
Arguments: none

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="120"/>
  <module name="status">
    <method name="getPlayerStatus"/>
  </module>
</interface>
```

Response
Module: status
Method: getPlayerStatus
Arguments: see the table below

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="120"/>
  <return id="0"/>
  <module name="status">
    <method name="getPlayerStatus">
      <arg name="playerState" value="Playing"/>
      <arg name="playlistName" value="Playlist 1"/>
    </method>
  </module>
</interface>
```

Response arguments table:

name	Range or possible values	Comment
playerState	Initializing	playlistName will be empty
	Playing	playlistName of running playlist will be set
playlistName	Playlist1	Name of the playlist, defined in your playlist.ddx file

3.5.2 Module: control

3.5.2.1 Method: activatePlaylistTrigger

Note: The Http header details are not listed for every XML call. A detailed description of the demanded http header is to be found in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**.

Supported since firmware version:

Product name	Firmware version
DigiPoster-III	n.a
DigiPoster-4.3	n.a.
VideoPoster-III	1.2.0

Description:
A playlist.ddx file can also contain triggers. These trigger events can be activated with the method activatePlaylistTrigger by transferring the triggerId. Setting a not existing triggerId will result in an XML error reply.

Request:
Module: control
Method: activatePlaylistTrigger
Arguments: see the table

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="12366"/>
  <module name="control">
    <method name="activatePlaylistTrigger">
      <arg name="triggerId" value="4"/>
    </method>
  </module>
</interface>
```

Response
Arguments: none

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="12366"/>
  <return id="0" message="Success"/>
  <module name="control">
    <method name="activatePlaylistTrigger">
      <!-- no arguments! -->
    </method>
  </module>
</interface>
```

Request arguments table:

name	Range or possible values	Comment
triggerId	Unsigned integer	Trigger Id must be defined in playlist

3.5.2.2 Method: playlistUploadDone

Note: The Http header details are not listed for every XML call. A detailed description of the demanded http header is to be found in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**.

Supported since firmware version:

Product name	Firmware version
DigiPoster-III	n.a
DigiPoster-4.3	n.a.
VideoPoster-III	1.6.0

Description:

Playlist upload done call is used to request start of playing the newly uploaded playlist after all file uploads complete.

Request
Module: control
Method: playlistUploadDone
Arguments: none

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="12366"/>
  <module name="control">
    <method name="playlistUploadDone"/>
  </module>
</interface>
```

Response
Arguments: none

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name=”httpctrl” version="1.0.0">
  <request id=”12366”/>
  <return id=”0” message=”Success”/>
  <module name="control">
    <method name="playlistUploadDone">
      <!-- no arguments! -->
    </method>
  </module>
</interface>
```

3.5.2.3 Method: playerOn

Note: The Http header details are not listed for every XML call. A detailed description of the demanded http header is to be found in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**.

Supported since firmware version:

Product name	Firmware version
DigiPoster-III	n.a
DigiPoster-4.3	n.a.
VideoPoster-III	n.a.

Description:
The method playerOn switches the media player on.

Request
Module: control
Method: playerOn
Arguments: none

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="12366"/>
  <module name="control">
    <method name="playerOn"/>
  </module>
</interface>
```

Response
Arguments: none

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name=”httpctrl” version="1.0.0">
    <request id=”12366”/>
    <return id=”0”/ message =”Success”>
  <module name="control">
    <method name="playerOn">
      <!-- no arguments! -->
    </method>
  </module>
</interface>
```

3.5.2.4 Method: playerOff

Note: The Http header details are not listed for every XML call. A detailed description of the demanded http header is to be found in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**.

Supported since firmware version:

Product name	Firmware version
DigiPoster-III	n.a
DigiPoster-4.3	n.a.
VideoPoster-III	n.a.

Description:
The method playerOff switches the media player off.

Request
Module: control
Method: playerOff
Arguments: none

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="12366"/>
  <module name="control">
    <method name="playerOff"/>
  </module>
</interface>
```

Response
Arguments: none

```
<?xml version="1.0" encoding="UTF-8"?>
<interface name="httpctrl" version="1.0.0">
  <request id="12366"/>
  <return id="0" message="Success"/>
  <module name="control">
    <method name="playerOff">
      <!-- no arguments! -->
    </method>
  </module>
</interface>
```

4 Appendix

Sample content of a minimal playlist.ddx file:

```
<?xml version="1.0" encoding="UTF-8"?>
<artistaProject version="1.4.0">
    <prjProp>
        <prjTitle>Project1</prjTitle>
        <prjTs>1329238002</prjTs>
    </prjProp>
    <prjCont>
        <device>
            <devProp>
                <devType>VIPO-III</devType>
                <devMinFwVer>1.0.0</devMinFwVer>
            </devProp>
            <devCont>
                <display>
                    <dispProp>
                        <dispOrient>0</dispOrient>
                        <dispRes>
                            <xRes>1920</xRes>
                            <yRes>1080</yRes>
                        </dispRes>
                    </dispProp>
                    <dispCont>
                        <playlist>
                            <plProp>
                                <plTitle>Playlist 1</plTitle>
                                <plLenms>15000</plLenms>
                                <plBrt>100</plBrt>
                          </plProp>
                            <plCont>
                                <item>
                                    <src>video1.mp4</src>
                                    <ts>1326288523</ts>
                                </item>
                            </plCont>
                        </playlist>
                    </dispCont>
                </display>
            </devCont>
        </device>
    </prjCont>
</artistaProject>
```

Our company network supports you worldwide with offices in Germany, Great Britain, Italy, Turkey and the USA. For more information please contact:

DATA DISPLAY GROUP

Distec GmbH
Augsburger Str. 2b
82110 Germering
Germany

Phone: +49 (0)89 / 89 43 63-0
Fax: +49 (0)89 / 89 43 63-131
E-Mail: info@datadisplay-group.de
Internet: www.datadisplay-group.de

Display Technology Ltd.
5 The Oaks Business Village
Revenge Road, Lordswood
Chatham, Kent, ME5 8LF
United Kingdom
Phone: +44 (0)1634 / 67 27 55
Fax: +44 (0)1634 / 67 27 54
E-Mail: info@datadisplay-group.co.uk
Internet: www.datadisplay-group.co.uk

Apollo Display Technologies, Corp.
87 Raynor Avenue, Unit 1Ronkonkoma, NY
11779
United States of America
Phone: +1 631 / 580-43 60
Fax: +1 631 / 580-43 70
E-Mail: info@datadisplay-group.com
Internet: www.datadisplay-group.com

Sales Partner:

REM Italy s.a.s.
di Michieletto Flavio & C.
Via Obbia Bassa, 10
I-35010 Trebaseleghe (PD)
Italy
Phone: +39 335 521 37 89
E-Mail: info@remitaly.com
Internet:www.remitaly.com

Sales Partner:

DATA DISPLAY BİLİŞİM TEKNOLOJİLERİ
İÇ VE DIŞ TİCARET LİMİTED ŞİRKETİ
Barbaros Mh Ak Zamabak Sk A Blok
D:143 Ataşehir/İstanbul
Turkey
Phone: +90 (0)216 / 688 04 68
Fax: +90 (0)216 / 688 04 69
E-Mail: info@data-display.com.tr
Internet:www.data-display.com.tr