AZURE DEVELOPMENT SOLUTIONS

NetworkTV Server API Guide

Phone: +44 (0) 1425 650697
E-Mail: info@azure-ds.com Web: www.azure-ds.com
Release 1.2

User Calls

This document contains a list of the application programming interfaces (API) published for the NetworkTV solution. It includes instructions for most elements of the NetworkTV solutions including the interaction with live streams, recording, managing content and managing set top boxes.

All user calls are fetched using HTTP GET

User Calls

Get XML List of Live Channels

http://111.67.19.54:8088/aurora/LiveChannels.xml"

Get XML List of On Demand Channels

http://111.67.19.54:8088/aurora/OnDemandChannels.xml"

Get XML List of On Demand Content

http://111.67.19.54:8088/aurora/Content.xml"

Get XML List of On Demand Tags

http://111.67.19.54:8088/aurora/tags.xml"

Get XML List of Set Top Boxes

http://111.67.19.54:8088/aurora/stbs.xml"

sample XML

```
<stbs>
  <stb name="Reception" id="86CE361A-6839-B833-0899-07F0AB7F553E" ip="192.168.0.3"
  status="Playing > Tracks : Adam Driver, Mia Wasikowska, Emma Booth">

   <groups>

    <group name="Ground Floor" id="B6DB3436-18C6-303F-F12F-0EEE86CF62E2"/>

   </groups>
  </stb>

</stbs>
```

Get XML List of Set Top Box Groups

http://111.67.19.54:8088/aurora/stbgroups.xml"

sample XML

```
<groups>

  <group name="Ground Floor" id="B6DB3436-18C6-303F-F12F-0EEE86CF62E2"/>

  <group name="First Floor" id="E9D25E6A-FFB6-B869-9CF5-13AC9151B119"/>

  <group name="Second Floor" id="324CBC54-04AF-EC1A-9A28-13B1A1C4C167"/>
</groups>
```

Management Calls

All management calls are sent using HTTP POST

Sample raw data from call to Move Live Channel Up the list

POST /aurora/managechannel HTTP/1.1
Host: localhost
Content-Length: 77
content-type: application/x-www-form-urlencoded
DNT: 1
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.154 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Pragma: no-cache
Cache-Control: no-cache

channelid=E5043DA1%2DE85C%2D4EA3%2DB97B%2DA735807D8B38&mode=movelivechannelup

Server response will either be an OK or ERROR or a json with Call, Status and a message.

Manage Channels Commands

http://111.67.19.54:8088/aurora/managechannel

POST

Move Live Channel Up the list

mode : movelivechannelup
channelid : <id>

POST

Move Live Channel Down the list

mode : movelivechanneldown
channelid : <id>

POST

Delete a Live Channel

mode : deletelivechannel
channelid : <id>

POST

Add a new Live Channel

mode : addlivechannel
channelid : <new channels id>
name : <new channels name>
type : <DEMO | H.264 | MPEG-2 | encoder name>
source : <url to multicast feed> //e.g. udp://@234.0.0.0:1234
username : <user name to log into supported encoder>
password : <password to log into supported encoder>
encoderip : <ip address of supported encoder>
reflect : Boolean, true or false if channels is reflected
destination: <url to multicast out reflected feed> //e.g. udp://@234.0.0.0:1234

Update a Live Channel

mode : updatelivechannel
channelid : <id>
name : <new name>
type : <DEMO | H.264 | MPEG-2 | encoder name>
source : <url to multicast or unicast (if reflected chan feed> //e.g. udp://@234.0.0.0:1234 or rtsp://@
username : <user name to log into supported encoder>
password : <password to log into supported encoder>
encoderip : <ip address of supported encoder>
reflect : Boolean, true or false if channels is reflected
destination: <url to multicast out reflected feed> //e.g. udp://@234.0.0.0:1234

Move an On Demand Channel Up the list

mode : moveondemandchannelup
channelid : <id>

Move **an On Demand Channel Down the list**

mode : moveondemandchanneldown
channelid : <id>

Delete an On Demand Channel

//NB you need to remove any references to this channel in the on demand content list

mode : deleteondemandchannel
channelid : <id>

Add a new On DemandChannel

mode : addondemandchannel
channelid : <new channels id>
name : <new channels name>

Update a Live Channel

mode : updateondemandchannel
channelid : <id>
name : <new name>

Start an HLS from a Live Channel

mode : starthls
channelid : <id> of the live channel for the parent of the HLS

Stop HLS - An HLS stream is automatically stopped and deleted if its parent channel is deleted

mode : stophls
channelid : <id> of the HLS channel

Recording

URL = http://111.67.19.54:8088/aurora/recording

Start a Recording Note metadata is added using updatemetadata and the poster frame is uploaded using upload.jsp

mode: set

recordingfrom: LiveChannel uuid to record from

recorderid: The uuidid of the recorder to use

recordingto: Requested to record to channel uuid, ie channel to tag it with in metadata

channelid: Channel uuid to record from

text: Text description (like in the normal video meta data)

title: Title of the recorded video file

starttime: When do you want this to start? (now, or a date/time)

duration: When do you want this to stop?

eg:

55 - 55 seconds

12:03:45 - 12 hours, 03 minutes and 45 seconds

usedby: Future use, which user is using the recorder

caption: Metadata caption, same as the video files (as this will end up as one)

// status is part of the obj, but is used to read the status of the recorder

// status: What is the recorder doing.... (started, recording, failed, stopped) unless better suggestions

mode: start

recordingfrom: LiveChannel uuid to record from

recorderid: the uuidid of the recorder to use

recordingto: Requested to record to channel uuid, ie channel to tag it with in metadata

channelid: channel uuidid to record from

// DEPRECATED ** recordingid: Becomes the video ID when finished

text: Text description (like in the normal video meta data)

title: Title of the recorded video file

starttime: When do you want this to start? (now, or a date time thingy)

duration: When do you want this to stop?

eg:

55 - 55 seconds

12:03:45 - 12 hours, 03 minutes and 45 seconds

usedby: Future use, which user is using the recorder

caption: Metadata caption, same as the video files (as this will end up as one)

// status is part of the obj, but is used to read the status of the recorder

status: What is the recorder doing.... (Started, Recording, Failed, Stopped)

Stop a Recording

mode: stop

recorderid : the id of the recorder to stop

Manage Content

URL = http://111.67.19.54:8088/aurora/upload.jsp

Upload / Replace a video (also used for uploading Poster Frames) N.B - uses multipart/form-data

mode : upload
videoid : <id of folder to upload the video to>
isfirstchunk : <true|false> //If 'true' and file exists then replace it
//If 'true' and file doesn't exist then it is the first (and possibly the only chunk)
//If 'false' then append this chunk to the end of the existing file
filechunk : <filename and file data>

Files which are larger than 3Mbytes are uploaded in 3MByte chunks

This works the same as a normal HTML multipart/form-data POST e.g.

```
<form id="form1" method="post" action="upload.aspx" enctype="multipart/form-data" >
  <input type="text" name="mode" value="upload" />
  <input type="text" name="videoid" value="123-456-769" />
  <input type="text" name="isfirstchunk" value="true" />
  <input type="file" name="filechunk" />
</form>
<button onclick="form1.submit()">Submit</button>
```

Server side C# code to show how it works

```
protected void Page_Load(object sender, EventArgs e)
{
    string uuid = Request.Form["videoid"];
    string isFirstChunk = Request.Form["isfirstchunk"];
    HttpPostedFile file = Request.Files["filechunk"];
    int fileSize = file.ContentLength;
    Byte[] fileByteArray = new byte[fileSize];
    file.InputStream.Read(fileByteArray, 0, fileSize);
    string path = "C:\\Aurora\\Data\\OnDemand\\" + uuid + "\\" + file.FileName;
    FileStream stream;
    if(isFirstChunk == "true")
{
if (File.Exists(path))
  File.Delete(path);
stream = new FileStream(path, FileMode.CreateNew, FileAccess.Write, FileShare.None);
}
```

```
                else
            {
           stream = new FileStream(path, FileMode.Append, FileAccess.Write, FileShare.None);
            }
        stream.Write(fileByteArray, 0, fileByteArray.Length);
        stream.Close();
            }
```

URL = http://111.67.19.54:8088/aurora/managecontent

Update / Add Video Metadata

mode : updatemetadata
videoid : <id>
title : <edited title>
Caption : <edited caption>
description: <edited description>
duration : <duration> e.g. 00:03:45

Add Tag to Video

mode : addtagtovideo
videoid : <video id>
tagid : <tag id>

Remove Tag from Video

mode : removetagfromvideo
videoid : <video id>
tagid : <tag id>

Delete one or more Videos

mode : deletevideo
videoids : <id,id,id...> //Comma seperated list of videos to delete

Delete an On Demand Tag

mode : deleteondemandtag //NB you need to remove any references to this tag in the on demand content list
tagid : <id>

Add a new On Demand Tag

mode : addnewondemandtag
tagid : <id>
name : <new tags name>

Add an On Demand Tag

mode : renameondemandtag

tagid : <id>

name : <new tags name>

Manage Set Top Boxes

URL = http://111.67.19.54:8088/aurora/managestbs

Move an STB Up the list

mode : movestbup

stbid : <id>

Move an STB Down the list

mode : movestbdown

stbid : <id>

Reboot

mode : reboot

stbid : <id>

Reboot All

mode : rebootall

Add STB

mode : addstb

stbid : id

ip : ip

name : name

Play Video On Demand

mode : playvod

stbid : ALL or a , seperated list of id's

groupid: A group to play to, if not ALL, can be blank if ALL or a specific STB

videoid: id of a video to play

Play Live Channel

mode : playlive

stbid : ALL or a , seperated list of id's

groupid: A group to play to, if not ALL, can be blank if ALL or a specific STB

channelid: id of a live channel to play

Stop

mode : stop

stbid : ALL or a , seperated list of id's

groupid: A group to play to, if not ALL, can be blank if ALL or a specific STB

Move STB UP

mode : movestbup

stbid : the ID of the stb

Move STB Down

mode : movestbdown

stbid : the ID of the stb

Delete

mode : delete

stbid : the ID of the stb

Delete group

mode : deletegroup

Groupid : ID of group to delete

Add group

mode : addgroup

groupid: I will generate this

groupname : name of the group

mcast : multicast address of the group as the STB's are assigned to listen to this multicast for commands

renamegroup // This is more of an update or edit group

mode : renamegroup

groupid: I will generate this

groupname : name of the group

mcast : multicast address of the group as the STB's are assigned to listen to this multicast for commands

Add STB to group

mode : addstbtogroup

groupid: the id of the group

Remove STB from group

mode : removestbfromgroup

groupid: the id of the group

Amino UI

In the Demo version the Amino is set to load its web page from:
http://111.67.19.54:8088/Amino.html

When the Amino UI starts up it waits a random interval between 1->3 seconds and then sends a:
GET http://111.67.19.54:8088/stbstatusupdate?status=Idle

The demo app then checks its stbs.xml file to see if the received stbstatusupdate comes from an unknown address.
If it does, it just updates the 'status' attribute from that IP address and if not it adds a new entry to the stbs.xml file.

The Amino UI adds an event handler for AminoGeneric.onEvent13. In the Amino docs this is the Emergency Alert System number. This is used by the demo app to send play commands to the STB. The server will use the stbremoteconf app to send the EAS string.

Play Live Channel
String sent using EAS
src=igmp://234.5.6.7:1236>NetworkTV Live
Params
src=igmp://234.5.6.7:1236 = URL to Play
NetworkTV Live = Channel Name

Play VOD
Command
src=http://192.168.0.101:8088/data/ondemand/581d37ab-f24c-cca9-801c-530cf309ed0d/video.mp4>Horrible%20Bosses%202%20@3A%20Jennifer%20Aniston@2C%20Chris%20Pine@2C%20Christoph%20Waltz

Params
src=http://192.168.0.101:8088/data/ondemand/581d37ab-f24c-cca9-801c-530cf309ed0d/video.mp4 = URL to Play
Horrible%20Bosses%202%20@3A%20Jennifer%20Aniston@2C%20Chris%20Pine@2C%20Christoph%20Waltz = Title>Caption

When the Amino receives a command it plays the URL and then when it receives an AVMediaEvent it calls back to the server to update its status e.g.

GET http://111.67.19.54:8088/stbstatusupdate?status=Playing > NetworkTV Live

Discrete UI API Commands

URL = http://<serverip>:8088/aurora/api

Get channels by uuid:

command : getVideosByChannel

channeluuid : channeluuid you are interested in