TechNote – AltitudeCDN™ Multicast+ Integration with TalkPoint

Version 1.1

AltitudeCDN™ Multicast+ is a patented (US Pat. 9,516,390) solution that brings multicast support to any live video deployment that uses HTTP Live Streaming (HLS) or DASH. Multicast+ integrates with the TalkPoint video event management service, providing TalkPoint customers with a solution that efficiently delivers high-quality live (or video on demand (VOD) published as live) streaming across the enterprise, without overloading or degrading network resources.

This guide describes how to provision and deploy a TalkPoint video event that uses AltitudeCDN Multicast+.

Contents

Introduction

During live (or video on demand (VOD) published as live) events, large numbers of viewers each attempt to connect separately to external internet content delivery servers, creating a high demand on network resources that can lead to poor video quality, or even service denial for event viewers and other network clients.

To avoid these issues, Ramp and TalkPoint have jointly developed a solution where TalkPoint customers can integrate AltitudeCDN Multicast+ into their video content management and delivery environment. By using Multicast+ Senders and Receivers, a video event is presented as a single multicast stream that is shared simultaneously by multiple viewers, allowing an event deployment to scale to large audiences, while conserving network bandwidth.

The TalkPoint event management interface supports parameters to provision Multicast+ events. Typically, TalkPoint configures these parameters for a customer as a managed service, however, the Multicast+-related management interface can be exposed for customer access. The on-premise Multicast+ Sender application must be configured separately to match the corresponding TalkPoint event, and the Multicast+ Receiver application must be pre-installed for all viewers. The TalkPoint video player and portal page is enabled to detect and receive Multicast+ streams if present, or fall back to unicast transmission if none is found.

Requirements

To use Multicast+ with TalkPoint, the following requirements exist:

Item	Recommendation
Multicast+ Sender	Multicast+ Sender v1.6.5 or later – The Sender must be installed and running on an on-premise Windows or CentOS platform, positioned within the customer network so multicast traffic can reach the audience. For more information, see the *AltitudeCDN Multicast+ Deployment Guide*.
Multicast+ Receivers	Any of the following: • Multicast+ Windows Receiver 1.8 or later. • Multicast+ Windows Service Receiver 1.0.47 or later. • Multicast+ Windows Add-On Receiver 1.1.25 or later. • Multicast+ Mac Receiver 1.6.5 or later. The receivers must be installed on each client viewing device prior to the event. For more information, see the *AltitudeCDN Multicast+ Deployment Guide*.

Solution Summary

Event viewers initiate playing the video event normally by accessing the TalkPoint Player page. The TalkPoint Player (or any other Multicast+-aware player) then searches for and detects a local Multicast+ Receiver that can receive the pre-configured multicast streams. If no Receiver or multicast stream is available, the TalkPoint Player falls back to unicast transmission.

The Multicast+ Sender uses a management URI that has been provided for the TalkPoint event. This URI supplies the Sender Agent with a dynamic configuration that determines which multicast addresses and ports are used for each stream. Any stream switching initiated by the presenter is reflected automatically on the viewer's player.

Note: *For information on configuring the Multicast+ Sender, see the AltitudeCDN Multicast+ Deployment Guide.*

To use Multicast+ with TalkPoint:

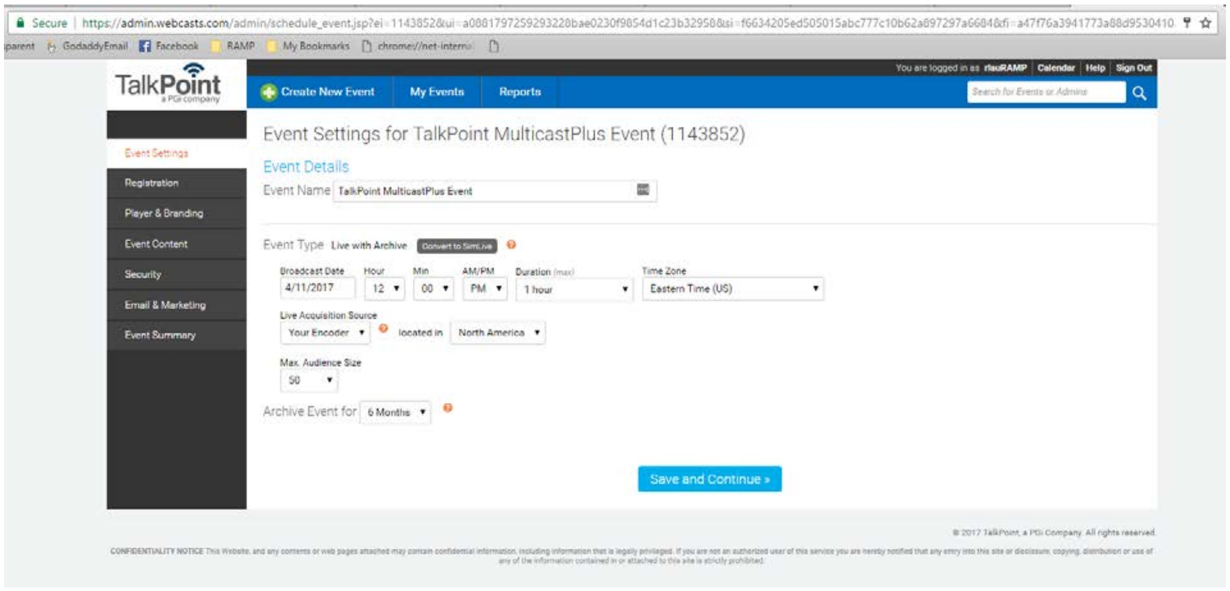
1. **(Typically configured by TalkPoint)** Create a matching Multicast+ key pair, then provision:
   - The player landing page for the specific customer.
   - The corresponding senderKeyStore in the customer's on-premise senderAgent.properties file.
2. Configure each TalkPoint video event stream (for example, Prelive, Primary, and Backup) with a unique multicast address and UDP port pair (for example, 239.1.1.1:3500).

   Note: *Multicast address/port pairs for events require unique multicast addresses to prevent overlap issues with other events.*
3. Configure the URI for the local Multicast+ Receiver (for example `http://talkpoint.multicast-receiver-altitudecdn.net:12345`, or `https://talkpoint.multicast-receiver-altitudecdn.net:12350`):
   - HTTPS is recommended to match the scheme of the player portal.
   - talkpoint.multicast-receiver-altitudecdn.net resolves to 127.0.0.1 in the public DNS.
4. TalkPoint provides a source URI that can be added to the senderAgent.properties file. For example:
   - `vdmsagent.managementUri=https://event.webcasts.com/viewer/ramppullpaths.jsp?ei=1143852&sh0=df909e2f4afd61de52eb8e89f3e886c7249c2294`
   - This source URI provides dynamic JSON-based configuration to the Multicast+ Sender agent to support stream switching.

TalkPoint Multicast+ Configuration

This section summarizes the configuration steps for using Multicast+ with TalkPoint (for more information on the TalkPoint interface, see TalkPoint documentation):

1. Create a new Event.
2. Within Event Settings, set the appropriate values for the Event Name, Event Type, and Archive Events settings. Note that only live events are supported by the TalkPoint Multicast+ integration.



Event Settings for TalkPoint MulticastPlus Event (1143852)

Event Details

Event Name: TalkPoint MulticastPlus Event

Event Type: Live with Archive

Broadcast Date: 4/11/2017
Hour: 12
Min: 00
AM/PM: PM
Duration (max): 1 hour
Time Zone: Eastern Time (US)

Live Acquisition Source: Your Encoder located in North America

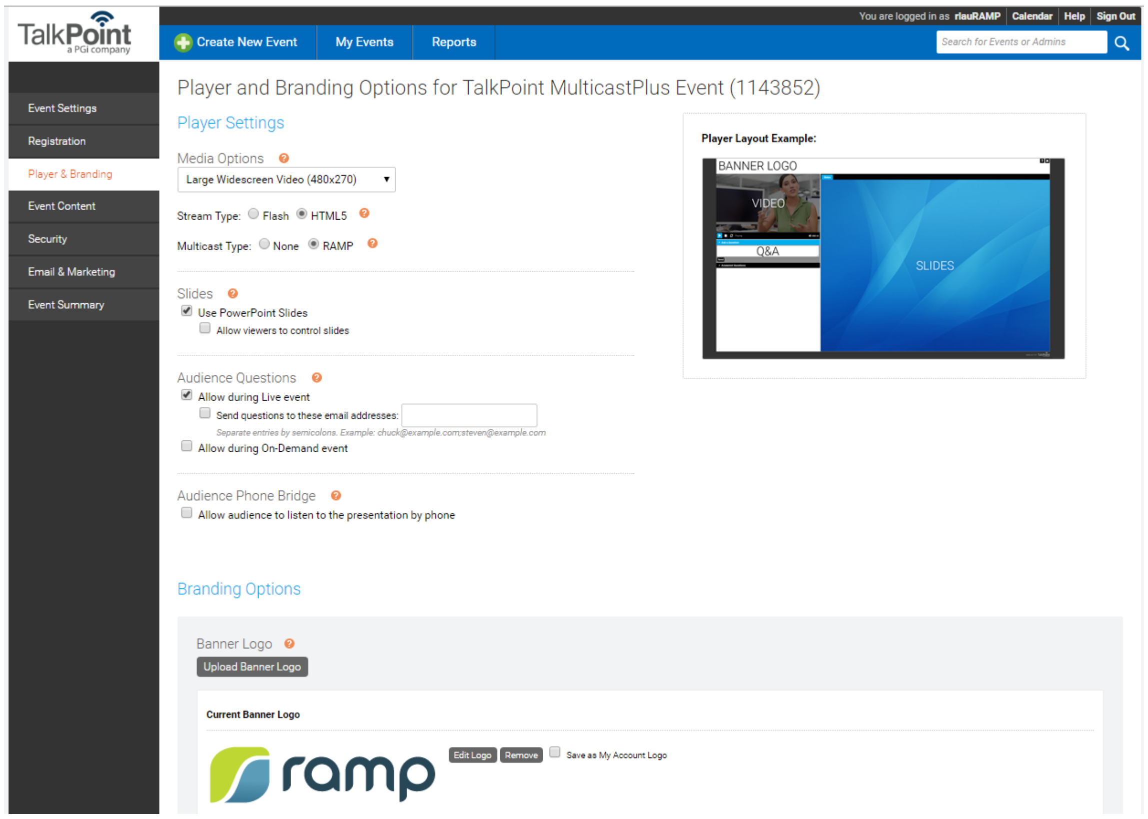
Max. Audience Size: 50

Archive Event for: 6 Months

© 2017 TalkPoint, a PGi Company. All rights reserved.

CONFIDENTIALITY NOTICE This Website, and any contents or web pages attached may contain confidential information, including information that is legally privileged. If you are not an authorized user of this service you are hereby notified that any entry into this site or disclosure, copying, distribution or use of any of the information contained in or attached to this site is strictly prohibited.

3. Within the Player and Branding Options:
   - For Media Options, select:
     - Stream Type: HTML5
     - Multicast Type: RAMP
   - Set the appropriate values for the Slides, Audience Questions, and Audience Phone Bridge settings.


You are logged in as rlauRAMP | Calendar | Help | Sign Out

Create New Event | My Events | Reports

Event Settings
Registration
Player & Branding
Event Content
Security
Email & Marketing
Event Summary

Player and Branding Options for TalkPoint MulticastPlus Event (1143852)

Player Settings

Media Options: Large Widescreen Video (480x270)

Stream Type: Flash [✓ HTML5]

Multicast Type: None [✓ RAMP]

Slides
[✓] Use PowerPoint Slides
[ ] Allow viewers to control slides

Audience Questions
[✓] Allow during Live event
[ ] Send questions to these email addresses:
Separate entries by semicolons. Example: chuck@example.com;steven@example.com
[ ] Allow during On-Demand event

Audience Phone Bridge
[ ] Allow audience to listen to the presentation by phone

Player Layout Example:

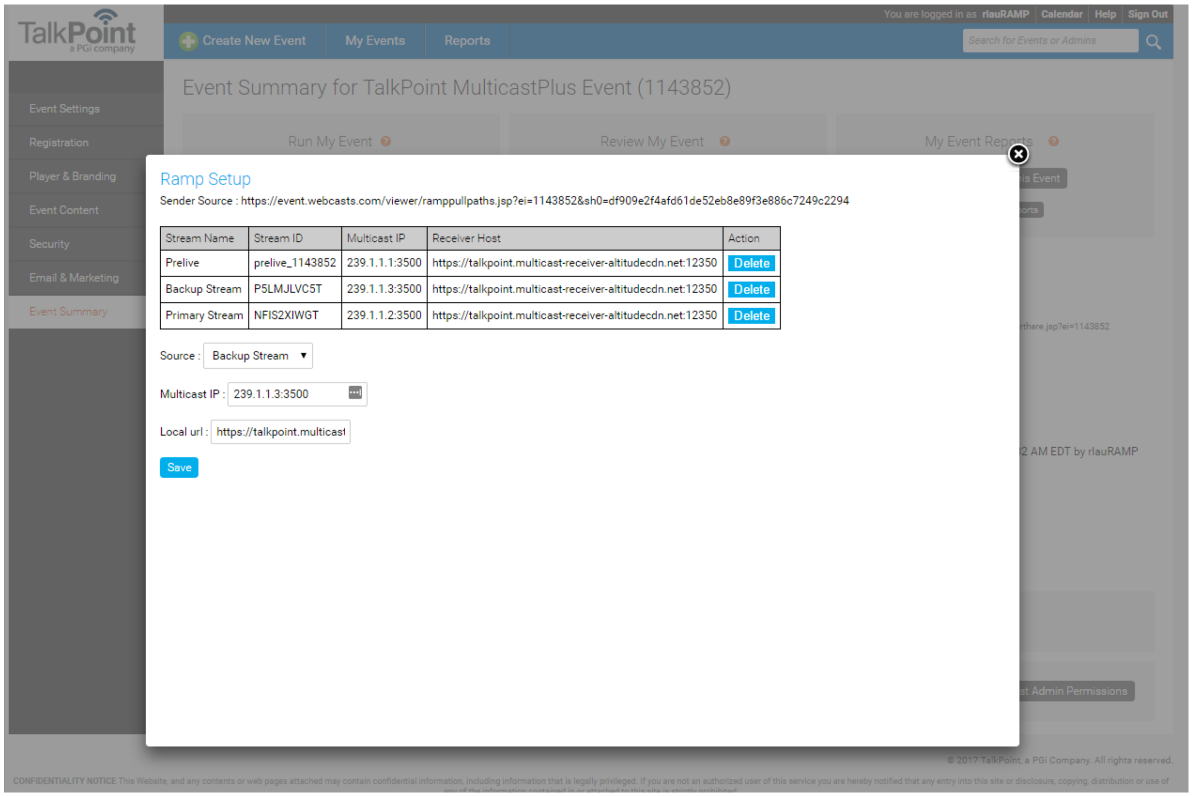
Branding Options

Banner Logo
Upload Banner Logo

Current Banner Logo
Edit Logo | Remove | [ ] Save as My Account Logo

4. Within Event Summary, display the Ramp Setup panel:
   - Define the Prelive, Primary, and Backup stream multicast IPv4 and ports used for the event. These addresses must be coordinated with the customer network IT to ensure that no collision or overlap occurs with other events or network traffic.
   - A default DNS name (talkpoint.multicast-receiver-altitudecdn.net => 127.0.0.1) is used to identify the local Multicast+ Receiver Host. This hostname can be used for all TalkPoint deployments.
     A valid PKI certificate for this hostname is embedded in the Multicast+ Receiver to allow HTTPS delivery of video content (for example: `https://talkpoint.multicast-receiver-altitudecdn.net:12350`).

     Note: *Ensure that the Receiver Host URL port matches the scheme used (for example, HTTP uses 12345, and HTTPS uses 12350).*
   - Copy the Sender Source URL to the Multicast+ Sender configuration. For example: vdmsagent.managementUri=https://event.webcasts.com/viewer/ramppullpaths.jsp?ei=1143852&sh0=df909e2f4afd61de52eb8e89fe886c7249c2294



Ramp Setup

Sender Source : https://event.webcasts.com/viewer/ramppullpaths.jsp?ei=1143852&sh0=df909e2f4afd61de52eb8e89f3e886c7249c2294

Stream Name	Stream ID	Multicast IP	Receiver Host	Action
Prelive	prelive_1143852	239.1.1.1:3500	https://talkpoint.multicast-receiver-altitudecdn.net:12350	Delete
Backup Stream	P5LMJLVC5T	239.1.1.3:3500	https://talkpoint.multicast-receiver-altitudecdn.net:12350	Delete
Primary Stream	NFIS2XIWGT	239.1.1.2:3500	https://talkpoint.multicast-receiver-altitudecdn.net:12350	Delete

Source : Backup Stream

Multicast IP : 239.1.1.3:3500

Local url : https://talkpoint.multicast

Save

Multicast+ Sender Agent High-Availability

The Multicast+ Sender Agent can be configured to support high-availability (HA) deployments, where clusters of Multicast+ Sender Agents provide redundancy for multicast channel video transmissions.

To configure high-availability, do the following on each Sender Agent that is a member of the cluster:

1. Download the file “ipaddr.min.js” from the following location and save it to the same directory as the vdms.properties file:

   `https://github.com/whitequark/ipaddr.js/blob/master/ipaddr.min.js`

2. Within the senderAgent.properties file, add the following:

Property	Description
vdmsagent.haCluster	The list of IPv4 addresses of the Sender Agents in the cluster. For example: `vdms.haCluster=<sender1>,<sender2>, ...`
vdmsagent.haPriority	The priority assigned to the current Sender Agent in the cluster. If an issue occurs with the active Sender Agent, the Sender Agent with the next highest priority resumes the transmission. Valid values: Integer, where a lower number denotes a higher priority. For example: vdmsagent.haPriority=*n* ***Note:*** *Priority assignments must be unique to clearly establish precedence.*
vdmsagent.haGroup	Identifies the media server to which the current Sender Agent is connected. For example, if you have two HLS servers, Sender(s)

	connecting to the first should be configured with haGroup=0 and to the second with haGroup=1. For more information, see the *AltitudeCDN Multicast+ Deployment Guide*. Valid values: 0-9 Default value: 0
vdmsagent. customProcessorJS	A JavaScript processing extension that allows you to define JavaScript code. For high-availability, the Example customProcessorJS Function shows how you can use customProcessorJS to create an adapter function that does the following: • Adds custom properties to make each event highly available between the Senders that belong to the cluster. • Select a unicast port based on the lower two octets of the multicast address.

Example customProcessorJS Function

An example of how the customProcessorJS property can be used for HA deployments is provided below:

```
# Start of JS Code
vdmsagent.customProcessorJS=\n\
load("ipaddr.min.js");\n\
var addrexp=new RegExp("([0-9\\.]+)",'g');\n\
function convert(agentProps,stream) {\n\
        var mobj = {\n\
                        "multicastAddress": stream.maddress + "/10",\n\
                        "sourceUrl": stream.url,\n\
                        "customProperties":{}, \n\
                        "D": agentProps.hasOwnProperty("D")? agentProps.D: 20, \n\
                        "sourceFormat": "hls"};\n\
        if (stream.type == "live"){ mobj.sourceType = "live"; }\n\
        else if (stream.type == "od") {\n\
                mobj.sourceType = "vod";\n\
                if (stream.hasOwnProperty("offset")) {\n\
                        mobj.timeOffsetMs = stream.offset * 1000; /* offset in MS */ \n\
                }\n\
        } else {\n\
                /* unhandled stream.type value. */ \n\
                return undefined;\n\
        }\n\
        /* Only active streams are sent.  Negative offset filter: */ \n\
        if (mobj.hasOwnProperty("timeOffsetMs") && mobj.timeOffsetMs < 0){\n\
                return undefined;\n\
        }\n\
        if (mobj.hasOwnProperty("multicastAddress")) {\n\
           var maddr=mobj["multicastAddress"];\n\
           if (maddr.match(addrexp)){\n\
              var parsed=ipaddr.parse(maddr.match(addrexp)[0]);\n\
              /* print("ip: " + parsed.octets); */\n\
              haCluster(agentProps,mobj,parsed);\n\
           }\n\
        }\n\
        return mobj;\n\
};\n\
function haCluster(agentProps, stream, addr) {\n\
  /* print("Called haCluster"); */\n\
  if (stream && addr) {\n\
    /* print("Use HA addr: " + addr); */\n\
    var props = stream.hasOwnProperty("customProperties") ? stream.customProperties : {};\n\
    if (agentProps.hasOwnProperty("vdmsagent.haCluster")) {\n\
      var port = (addr.octets[2]<<8) | addr.octets[3];\n\
      /* print("Use port: " + port); */\n\
      var ha = agentProps["vdmsagent.haCluster"].split(",");\n\
      for (var i = 0; i < ha.length; i++){\n\
        ha[i] = ha[i].trim() + ":" + port;\n\
      }\n\
      /* print("HA port: " + port + " group: " + ha.join(","));*/\n\
      props["haClusterUnicastAddresses"] = ha.join(",");\n\
      props["haClusterUnicastListenerPort"] = port;\n\
    }\n\
    if (agentProps.hasOwnProperty("vdmsagent.haPriority")) {\n\
      props["haPriority"] = agentProps["vdmsagent.haPriority"];\n\
    }\n\
    if (agentProps.hasOwnProperty("vdmsagent.haGroup")) {\n\
      props["haGroup"] = agentProps["vdmsagent.haGroup"];\n\
    }\n\
    stream["customProperties"]=props;\n\
  }\n\
};\n\
function addChannels(agentProps,incoming,req) {\n\
        if (incoming.hasOwnProperty("streams")) {\n\
                var allChan = {};\n\
                for (var property in incoming.streams) {\n\
                        if (incoming.streams.hasOwnProperty(property)){\n\
                                var stream = incoming.streams[property];\n\
                                var mobj = convert(agentProps,stream);\n\
                                if (mobj !== undefined) {\n\
                                        allChan[stream.streamid] = mobj;\n\
                                }\n\
                        }\n\
                }\n\
                if (Object.keys(allChan).length > 0) { req.channels = allChan; };\n\
        }\n\
};\n\
function initRequest(agentProps, orig){\n\
  var req = {};\n\
  req.senderKeyStore = agentProps.senderKeyStore;\n\
  if (orig.hasOwnProperty("senderKeyStore")) {\n\
    req.senderKeyStore = agentProps.senderKeyStore;\n\
  }\n\
  return req;\n\
};\n\
function request(body,bodyStr,agentProps,env) {\n\
   if (bodyStr){\n\
     var incoming;\n\
     try {\n\
      incoming = JSON.parse(bodyStr);\n\
     } catch(e) {\n\
      print(bodyStr);\n\
      return {};\n\
     }\n\
     var outgoing = initRequest(agentProps, incoming);\n\
     /*\n\
     print ("--------------------------------------------------------");\n\
     print("INIT: " + JSON.stringify(outgoing, null, '    '));\n\
     print("INCOMING REQUEST: " + JSON.stringify(incoming, null, '    '));\n\
     print("AgentProperties: " + JSON.stringify(agentProps, null, '    '));\n\
     */\n\
     addChannels(agentProps,incoming,outgoing);\n\
     /*\n\
     print("OUTGOING REQUEST: " + JSON.stringify(outgoing, null, '     '));\n\
     print ("--------------------------------------------------------");\n\
     */\n\
     return outgoing;\n\
   }\n\
   return {};\n\
}
#  End of JS Code
```

Diagnostic Usage Statistics

To enable Multicast+ Sender diagnostics, you can run an additional instance of a Sender on the same platform as the Sender Agent by doing the following:

1. Add the following minimal configuration to the Diagnostics Server's properties file:
   ```
   # port both to receive data and serve http
   diagnostics.port=18000
   diagnostics.regionsFile=regions.json
   diagnostics.password=abc123
   diagnostics.noAuth
   ```
2. Start the Diagnostics Server using the following command:
   ```
   # nohup ./sender.sh diagnostics.properties > /dev/null 2>&1 &
   ```