Triton Digital On-Demand Advertising Guide 1.3.8

TRITON DIGITAL

Publication Information

1440 Ste-Catherine W, Suite 1200
Montreal QC H3G 1R8
Canada

514-448-4037
www.tritondigital.com

Document Version

Triton Digital On-Demand Advertising Guide 1.3.8

November 26, 2014

Confidentiality Statement

Disclaimer Notice

Customer Support

supportdesk@tritondigital.com

1-866-448-4037 extension 1

Contents

Document Change Log

Version	Date	Change
1.3.8	Nov. 26, 2014	• Removed deprecated Android Device ID description.
1.3.7	Nov. 6, 2014	• Added “Appendix: Using a2x.”
1.3.6	Sept. 12, 2014	• Changed references to “Campaign Manager” to “Tap.” Version number unchanged.
1.3.6	August 7, 2014	• Additional banner sizes added. See “Banner Capabilities” on page 18.
1.3.5	June 18, 2014	• Minor edits and correction of typos.
1.3.4	April 24, 2014	• Clarified requirements for demographic targeting query parameters. (See page 17.)
1.3.3	March 19, 2014	• Correction to the note added to the location targeting information added in version 1.3.2.
1.3.2	March 13, 2014	• Added two new location targeting attributes: postalcode and country. See page 16.
1.3	March 6, 2014	• Clarification of the placement of the IdSync script. See “Integrating Cookie Synchronization into the Player” on page 10.
1.3	Feb. 13, 2014	• Revised: Asset Constraint parameters on page 14 • New: Player Targeting/Banner Capabilities on page 18 • New: VAST-JSONP Support on page 21
1.2.1	Jan. 31, 2014	• Revised: table of supported Listener IDs. See “Listener ID Management” on page 7. • Revised: “iOS Identifier for Advertising (idfa)” on page 8. • New: “Google Advertising ID (gaid)” on page 8. • Revised: former Android Device IDs are now considered “legacy.” • New: “Application-Generated Listener ID (app)” on page 9.

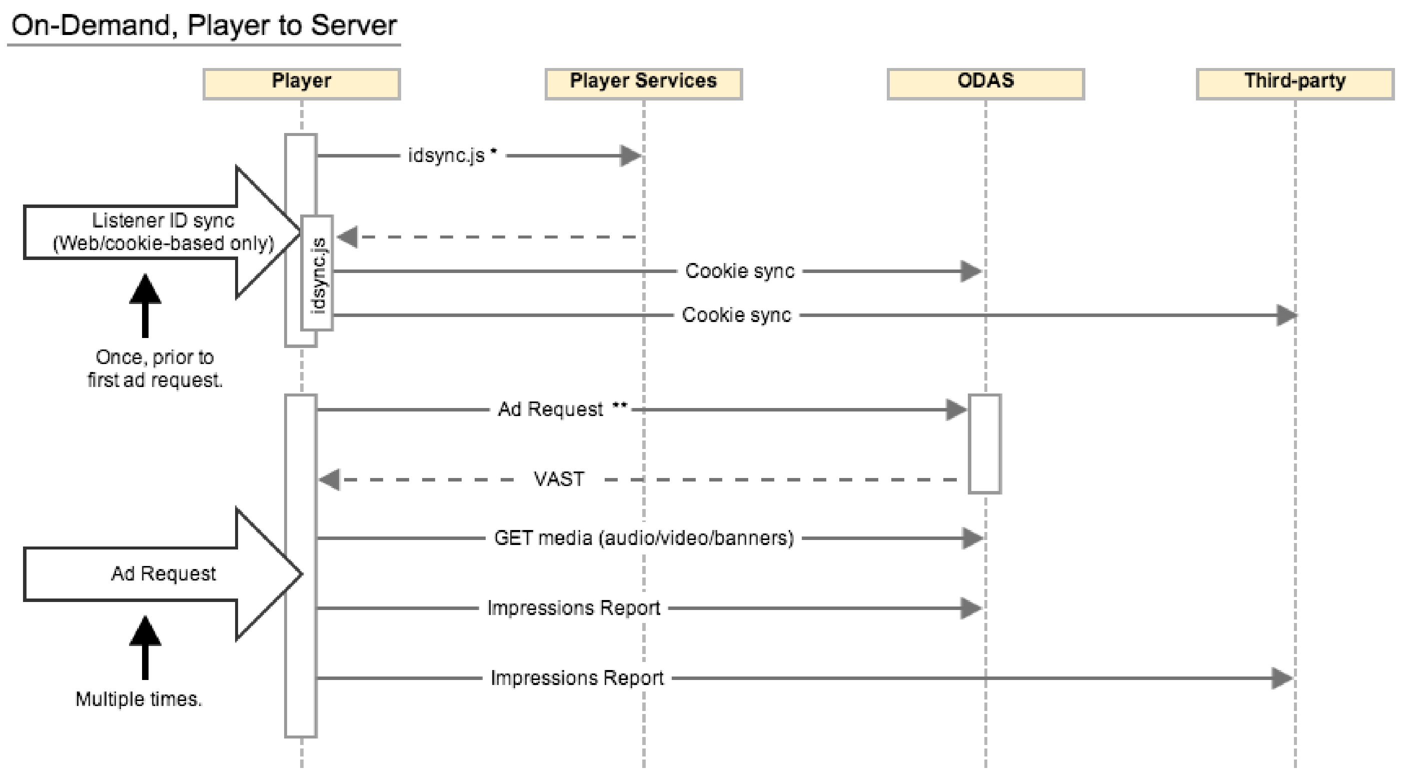
1. On-Demand Ad Service

The Triton Digital On-Demand Ad Service allows an application to request ads targeted to a given listener. In contrast with in-stream ads, these requests do not require a streaming connection, and they can request ads in a stand-alone fashion.

The On-Demand Ad Service is currently able to serve audio and video pre-rolls and mid-rolls from Triton Digital Ad Platform 3 and the Triton Digital a2x ad exchange. This document reflects the generic interface, but includes references to Ad Platform 3-specific aspects where applicable.

The On-Demand API supports both client-to-server (an application running directly on a listener's device, such as a smartphone, computer or other) and server-to-server (streaming server, media server or other) usage scenarios. This document primarily describes the client side API. See section 5 for information about server-to-server usage.

1.1 Flow Diagram: Player to Server



* **idsync.js:** Contains all third-party ad servers and DMPs.

** **Ad Request:**
- Device ID (for mobile/non-cookies)
- Targeting attributes
- Asset constraints
- Type (pre-roll/mid-roll)

2. Listener ID Management

Listener IDs (also called UUIDs) allow the ad platform to measure listenership of on-demand programs, and enables frequency capping and targeting of individual listeners. The types of Listener IDs used vary according to the client device/platform.

Triton supports the following types of Listener IDs (listed below in order of priority):

Type	Description	Usage
`idfa`	iOS identifier for advertising	iOS 6+ devices
`gaid`	Android advertising ID	Android applications
`wpdid`	Windows Phone Device ID	Windows Phone applications
`didsha1`	SHA1 hashed device ID	Legacy Android applications (deprecated; use `gaid` instead)
`didmd5`	MD5 hashed device ID	Legacy Android applications (deprecated; use `gaid` instead)
`macsha1`	SHA1 hashed MAC address	Legacy Android applications (deprecated; use `gaid` instead)
`openudid`	OpenUDID device ID	Legacy iOS devices (deprecated since iOS 6; use `idfa` instead)
`cookie`	Cookie-based Listener ID	Web-based players (i.e., Flash or HTML5)
`app`	Application-generated Listener ID	Players that do not have access to a Device ID, nor cookie support

Listener IDs are formatted as a prefix describing the type, a colon, and the value of the Listener ID:

```
<type>:<id>
```

For example:

```
didmd5:3b872b18b97bea478e9e5c6f85f6e48f
```

Some devices may support multiple types of IDs. The Triton Digital Streaming and On-Demand Ad Platforms only support a single Listener ID, so players must only select one from the table above (which is listed in order of priority – use the first one supported by the device/platform on which the player is running). If implementing server-to-server requests, see "Server-to-Server Cookie Synchronization" on page 23 for additional considerations.

2.1 Device-based Listener IDs

When developing a player application on a mobile device (or other non-Web-based platform), it is best to use the Device ID facilities provided by the device or its operating system. This Device ID is used as the Listener ID when interacting with Triton's platform.

See the Listener ID Management table on page 7 to determine the Listener ID type appropriate for your device/platform. The format of the value must be sent as returned by the device's OS.

The application must provide a single Listener ID as the `lsid` query parameter when calling the On-Demand Ad Request Service (as described in "Ad Requests" on page 12).

2.1.1 iOS Identifier for Advertising (idfa)

On iOS 6+ devices, use the `ASIdentifierManager` API to retrieve the IDFA, as follows:

```
NSString *idfaString = [[[ASIdentifierManager sharedManager]
advertisingIdentifier] UUIDString];
```

Note that the user can opt-out from the iOS Identifier for Advertising, so the application must handle cases where the ID is not available. In these cases, the application should fall back to cookie-based or application-generated Listener IDs, as described below.

2.1.2 Google Advertising ID (gaid)

Android applications should use the Google Advertising ID, available via Google Play Services. Documentation on retrieving and using this ID is available from: http://developer.android.com/google/play-services/id.html

Note that the user can opt-out from the Google Advertising ID, so the application must handle cases where the ID is not available. In these cases, the application should fall back to cookie-based or application-generated Listener IDs, as described below.

Note: Applications that cannot use Google Play Services (e.g., Amazon Kindle, or applications not hosted on the Google Play Store) should use one of the Legacy Android Device IDs described in the next section.

2.1.3 Windows Phone Device ID (wpdid)

On Windows Phone, the Device ID is retrieved using `DeviceExtendedProperties`, using code similar to:

```
var deviceId = string.Empty;

if (DeviceExtendedProperties.TryGetValue("DeviceUniqueId", out deviceUniqueId))
{
  var deviceIDbyte = (byte[])deviceUniqueId;
  deviceId = Convert.ToBase64String(deviceIDbyte);
}
```

More information is available from: http://msdn.microsoft.com/en-us/library/windowsphone/develop/microsoft.phone.info.deviceextendedproperties%28v=vs.105%29.aspx

2.1.4 OpenUDID Device ID (openudid)

OpenUDID is considered deprecated, and should not be used by new players. More information on OpenUDID is available from: https://github.com/ylechelle/OpenUDID

2.1.5 Application-Generated Listener ID (app)

For applications that cannot use one of the Device IDs listed above (either due to lack of platform support, or because of user opt-out) and are unable to support cookie-based Listener IDs, it is possible for the application to generate a Listener ID itself.

An application-generated Listener ID is a Type 4 GUID, as defined by the IETF standard RFC4122 (http://www.ietf.org/rfc/rfc4122.txt) and encoded as a lowercase character string (e.g., `550e8400-e29b-41d4-a716-446655440000`).

Once generated, the application-generated Listener ID must be preserved and reused as long as possible. At a minimum, it must remain the same for the duration of a listening session, but should ideally be persisted for as long as is practical (i.e., stored on-disk). It must **not** be regenerated on each Ad Request, as this defeats the purpose, and some features will not work as expected (e.g., frequency capping, unique listener metrics, etc.).

2.2 Cookie-based Listener IDs

Cookie-based Listener IDs are intended for Web-based and custom players. Mobile players should use one of the device identifier Listener ID types instead.

The cookie value’s format follows the IETF standard RFC4122 (http://www.ietf.org/rfc/rfc4122.txt) and is encoded as a lowercase character string (e.g., `550e8400-e29b-41d4-a716-446655440000`).

When a player connects for the first time it won’t have a Listener ID, so the On-Demand Ad Service assigns one to it. The Listener ID cookie is sent back to the player, which should save it and reuse it in subsequent connections. For Web-based players, the Web browser does this automatically. Custom player applications, however, might need to implement their own cookie handling.

2.2.1 How Cookie Synchronization Works

Prior to requesting ads that will be displayed in a browser-based application, it is the responsibility of the requesting application to ensure the browser has called the cookie synchronization functions.

A script inserted into the player page calls a number of 1x1 GIF files (“pixels”) that are inserted into the browser pane containing the player. These cookie synchronization pixels call a service at Triton, which redirects (returns a `302 Found` HTTP response) to the ad partner’s service. In response, the partner service redirects back to Triton Digital Player Services service with that service’s user ID. The Triton Digital service then returns the cookie synchronization pixel and writes the Triton cookie, containing the partner’s user ID.

Since the partner's user ID is now revealed to the Triton Ad Platform, it can send ad requests with that user ID, enabling targeted advertising from the partner's ad service.

2.2.2 Integrating Cookie Synchronization into the Player

The cookie synchronization process requires support for the cookie-based Listener IDs, as described above.

Cookie synchronization pixels are wrapped into a JavaScript that must be on the same page or window (or "pane") as the player. If you use a pop-out player, the script must be on the pop-out pane, not on the originating browser page.

To load the cookie synchronization script, insert one of the following HTML tags into the player window.

Option 1 – when you know the station name:

```
<script type="text/javascript"
src="http://playerservices.live.streamtheworld.com/api/idsync.js?station=ABCFM"/>
```

(… where ABCFM is the name of the station.)

Option 2 – when you don't know the station name but you know the mount name:

```
<script type="text/javascript"
src="http://playerservices.live.streamtheworld.com/api/idsync.js?mount=ABCFMAAC"/>
```

(… where ABCFMAAC is the name of the mount.)

Important! The script must be loaded into the player before the player requests ads from the On-Demand Ad Service. We recommend the script be inserted above the code that calls the player. *Be sure to put the script in the <body> of the page, not the <head>*.

Important! The cookie synchronization script described above supersedes all preceding cookie synchronization mechanisms described in previous versions of this guide. If one of the following URLs are in use, you should upgrade your player to the new cookie synchronization mechanism as soon as possible:

http://ib.adnxs.com/getuid?http://playerservices.live.streamtheworld.com/api/a2x.gif?partner=an&uid=$UID
http://playerservices.live.streamtheworld.com/api/uidsync
http://assets.streamtheworld.com/js/dmp.php
http://assets.streamtheworld.com/js/a2x.php

***Note**: Previous versions of the cookie synchronization mechanism are deprecated and will be removed*.

2.3 Hybrid Players

Players that use both In-Stream (i.e., Triton Digital-hosted streams) and On-Demand ads (via the On-Demand Ad Platform) must use the same Listener ID for both cases. Otherwise, some things like frequency caps and analytics will be affected and might not work as intended. Implementing the Listener ID according to the guidelines ensures it works correctly.

For more information, see the *Triton Digital Streaming Guide*.

2.4 Listener Tracking

We recommend that players support the Triton Digital Listener Tracking service. This is required in order to get advertising avails when using Tap with the On-Demand Advertising service.

If you are not using CM3, the Listener Tracking Service enables tracking of the player by Triton's Measurement Platform. This means that avails will be available immediately if you start using CM3, instead of having to wait 28 days for sufficient listening history to be accumulated. For more information about the Listener Tracking API, contact your Triton Digital solution specialist.

3. Ad Requests

3.1 HTTP Entry Point

The Ad Request Service is available over a public HTTP entry point:

```
http://<ondemand-ad-service-host>/ondemand/ars?type=<ad-type>&
fmt=<render-format>&(stid=<station-id>|stn=<station-name>)
[&lsid=<listener-id>]
```

A complete example, using many of the available parameters (Listener ID, Targeting Attributes, and Asset Constraints) would look like this:

```
http://ondemand.live.streamtheworld.com/ondemand/ars?type=midroll&fmt=vast&stn=D
EMOFM&lsid=idfa:70022383-d877-410b-bab2-e9c51c3854c1&lat=45.5088400&long=-
73.5878100&mindur=15&maxdur=30
```

3.2 On-Demand Ad Service Host

Please contact Triton Digital Customer Support to obtain the correct On-Demand Ad Service Host name to use for on-demand ad requests for pre-production and production uses.

3.3 Query Parameters

Parameter	Description	Value	Required
`type`	Ad type	String. Valid values: • `preroll` • `midroll`	Yes.
`fmt`	Rendering format	• `vast` • `vast-jsonp`	No. (Default is `vast`)
`stid`	Station ID	Integer value	Yes. (See note, below)
`stn`	Station Name	String	Yes. (See note, below)
`lsid`	Listener ID	String	No. (Strongly recommended if `uuid` cookie is not present. For more information, see “Listener ID Management” on page 7.)

Note: at least one of the `stid` or `stn` parameters is required.

3.3.1 Station IDs and Names

Either the Station ID or station name must be specified when calling the On-Demand Ad Request Service. While both IDs and names are supported, it is strongly recommended that clients use **station names**. If both ID and name are provided, the name is used (there is no validation check that the ID matches the name).

Triton Digital assigns station IDs and names when setting up a station. Station names are case-sensitive.

3.3.2 Listener IDs

As mentioned in the Listener ID Management section above, players must implement Listener ID support. The `lsid` parameter is not required; if it is absent, a cookie-based Listener ID is generated for the listener. If using cookies (which is recommended for non-mobile devices), then the `lsid` parameter may be omitted. However, if the player does not support cookies, the `lsid` parameter is required. If the player supports cookies, but does not yet have the cookie set, a new Listener ID cookie will be generated, and returned to the player as documented in "Listener ID Management" on page 7.

! **Warning:** It is required that players provide either a Listener ID via cookies or the `lsid` parameter, for performance and targeting effectiveness reasons.

3.3.3 Ad Types

When requesting an ad, it is required that the client specify what **type** of ad should be returned.

The available types are:

Ad Type	Description
`preroll`	Pre-roll ads. These are intended to be played before playback starts.
`midroll`	Mid-roll ads. These are intended to be played during the session, between content elements such as songs.

Depending on the station's configuration, different ad sources may be used to fulfill the ad request, such as Tap, a2x, or programming fallback. Contact Triton Digital to get information on the available ad sources, and to have your station configured to enable the desired content sources.

3.3.4 Rendering Formats

Supported render formats for the On-Demand Ad Service:

fmt parameter	Content-Type	Description
`vast`	`text/xml; charset=UTF-8`	VAST 2.0 (see section 4.1 for details)
`vast-jsonp`	`text/javascript; charset=UTF-8`	VAST 2.0 in a JSONP wrapper (see "JSONP Wrapper" on page 21 for details)

If no format is specified, `vast` is used. To prevent confusion, however, it is recommended that the rendering format be explicitly specified.

3.3.5 Asset Constraints

The On-Demand Ad Service provides the capability for players to specify constraints on the asset that can be returned.

Note: constraints only apply to the main creative, not to companion ads. When using the VAST rendering format, all companion ads are present in the VAST XML, and the client is responsible for selecting which ads it is able to display properly based on the included information.

Constraint	Query Parameter	Value	Audio	Video
Asset Type	`at=<list-of-asset-types>`	Comma-separated list of: • `audio` • `video`	N/A	N/A
Min/Max File Size	`minsz/maxsz=<size-in-kb>`	Integer (size in KB)	Yes	Yes
File Format (i.e. container)	`cntnr=<list-of-containers>`	Comma-separated list of: • `mp3` • `adts` • `flv` • `mp4`	Yes	Yes
Min/Max Duration	`mindur/maxdur=<seconds>`	Integer (duration in seconds)	Yes	Yes
Min/Max Bitrate	`minbr/maxbr=<bitrate-kbps>`	Integer (kbps)	Yes	Yes
Min/Max Width	`minw/maxw=<pixels>`	Integer (in pixels)	No	Yes

Constraint	Query Parameter	Value	Audio	Video
Mix/Max Height	`minh/maxh=<pixels>`	Integer (in pixels)	No	Yes
Audio: Codec	`acodec=<list-of-audio-codecs>`	Comma-separated list of: • `mp3` • `aac_hev1` • `aac_hev2` • `aac_lc`	Yes	Yes
Audio: Min/Max Channels	`minach/maxach=<channels>`	Integer	Yes	Yes
Audio: Sample Rates	`asr=<list-of-samplerates>`	Comma-separated list of sample rates (in Hz)	Yes	Yes
Video: Codec	`vcodec=<list-of-video-codecs>`	Comma-separated list of: • `h264` • `on2_vp6`	No	Yes
Aspect Ratio	`vaspect=<list-of-ratios>`	Comma-separated list of: • `4:3` • `16:9` • `other`	No	Yes
Min/Max Frame Rate	`minfps/maxfps=<frame-rate>`	Float (frames-per-second)	No	Yes

For example, if your player only supported MP3 and AAC-LC audio ads up to 30 seconds, you would append the following query parameters to the request shown in section 3.1 (HTTP Entry Point):

```
<ad-request-url>&at=audio&cntnr=mp3,adts&acodec=mp3,aac_lc&maxdur=30
```

Notes: *We strongly recommend not using constraints!* If it is unavoidable, the player should specify the minimum possible constraints, and only in cases where it considers some of the formats as incompatible. It is not advised that (for example) an exhaustive list of bit rates be specified if the player can accept any bit rate. *The more constraints are specified, the smaller the set of available ads.*

Also note that built-in constraints are compatible with all Flash-based players, so constraints are mostly applicable to custom clients such as mobile applications.

Finally, note that some constraints do not apply to all media types, and will be ignored. For example, specifying an image format when requesting a video ad will have no effect.

3.4 Listener Targeting

The Triton Digital Ad Platform's ad targeting capability uses the player's HTTP User Agent string, as well as GeoIP lookups (i.e., the listener's location is determined from his IP address) to establish the listener's basic targeting profile. Players may also append additional targeting attributes to the On-Demand Ad Request's query parameters, as detailed below.

Note: The targeting query parameters described below are used as follows:

- For **streaming** clients/players: targeting attributes are appended to the **Stream URL's query parameters**.
- For **on-demand** clients/players: targeting attributes are appended to the **On-Demand Ad Request's query parameters**.

3.4.1 Location Targeting

Below is the list of officially supported Location Targeting query parameters that clients/players may specify in On-Demand Ad Requests.

Query Parameter	Description	Valid Values	Required
`lat`	Latitude	Floating-point value: -90.0 to 90.0	No*
`long`	Longitude	Floating-point value: -180.0 to 180.0	No*
`postalcode`	Postal/ZIP code	Valid postal or ZIP code, without spaces. E.g., 89040 or H3G1R8.	No †
`country`	Country Code	ISO 3166-1 alpha-2 two-letter country code (e.g., US)	No †

* *Not required individually, but if you specify one, you must specify both*.

† *Not required individually. If using, however, we recommend that you specify both*.

When specifying **latitude/longitude**, clients/players should use the highest precision that is reasonable, depending on the device, platform, and battery usage considerations. (We suggest precision of 100 meters.)

Note: currently, the `lat` and `long` parameters only work with Tap and **a2x**-based advertising. It will function across the entire Ad Platform in an upcoming deployment.

3.4.2 Demographic Targeting

Below is the list of officially supported Demographic Targeting query parameters that clients/players may specify in On-Demand Ad Requests.

Query Parameter	Description	Valid Values	Required
`age`	Age	Integer value: 1 to 125	No*
`dob`	Date of Birth	String formatted as YYYY-MM-DD	No*
`yob`	Year of Birth	Integer value: 1900 to 2005	No*
`gender`	Gender	“`m`” or “`f`” (case-sensitive)	No

**Clients/players must specify only one of Age, Date of Birth, or Year of Birth.*

Note: Demographic targeting query parameters only work with Tap advertising, and only if demographic targeting is enabled for your broadcaster. Contact Triton Digital Customer Support to enable demographic targeting.

3.4.3 Custom Segment ID Targeting

Broadcasters that want to differentiate their listeners into custom broadcaster-specific segments may use the Custom Segment Targeting capability of Tap.

Below is the officially supported Custom Segment ID Targeting query parameter that clients/players may specify in On-Demand Ad Requests.

Query Parameter	Description	Valid Values	Required
`csegid`	Custom Segment ID	Integer value: 1 to 1000000	No

Note: Before use by players, please contact the Triton Digital Support Team to enable Custom Segment ID Targeting for your broadcaster. Currently, Custom Segment ID Targeting only works with Tap advertising.

3.5 Player Targeting

3.5.1 Banner Capabilities

Players can provide details on their level of support for banners, such as banner sizes and formats.

Query Parameter	Description	Valid Values	Required
`banners`	Banner Capability Flags	Comma-separated list of supported banner capabilities	No

The following capabilities are supported:

Capability	Description
`970x250`	IAB Billboard (970x250)
`120x60`	IAB Button 2 (120x60)
`300x600`	IAB Half Page/Filmstrip (300x600)
`728x90`	IAB Leaderboard (728x90)
`970x100`	IAB Leaderboard (970x100)
`300x250`	IAB Medium Rectangle (300x250)
`88x31`	IAB Microbar (88x31)
`300x1050`	IAB Portrait (300x1050)
`970x90`	IAB Pushdown (970x90)
`180x150`	IAB Rectangle (180x150)
`320x480`	IAB Smartphone Portrait (320x480)
`300x50`	IAB Smartphone Static Banner (300x50)
`320x50`	IAB Smartphone Static Wide Banner (320x50)
`300x300`	IAB Square (300x300)
`970x66`	IAB Super Leaderboard (970x66)
`160x600`	IAB Wide Skyscraper (160x600)
`Client-defined (w x h)`	Custom banner size
`swf`	Flash (SWF in HTML IFrame)
`vpaid`	VPAID

For example, a player that supports IAB Leaderboard and Medium Rectangle banners, in SWF format, would send the banners query parameter as follows:

```
banners=728x90,300x250,swf
```

The ordering of the capability flags is not important.

> **Note**: Before attempting to use player capability targeting, please contact the Triton Digital Support Team to enable Player Capability Targeting for your broadcaster. Currently, Player Capability Targeting only works with Tap advertising.

3.6 HTTP Response Codes

Code	Message	Description
`200`	`OK`	Success.
`204`	`No Content`	Returned if no ad is available (except for VAST format, which returns an empty VAST XML – see section 4.1.1, “Empty Response.”)
`404`	`Not Found`	The Ad Type or Station ID/Name provided does not exist.
`406`	`Not Acceptable`	The Rendering Format specified is invalid.
`500`	`Internal Server Error`	Failure.

> **Note**: other standard HTTP errors, such as `400 Bad Request`, or `405 Method Not Allowed`, may also be returned. Applications should handle HTTP response codes according to standard conventions, as described in w3.org protocol RFC2616, Section 10.

4. Ad Responses

4.1 VAST 2.0 Ads

The VAST rendering returned by the On-Demand Ad Service is VAST 2.0.1 compliant. The client application should thus support the IAB's VAST 2.0 Specification.

The VAST subset that **must** be supported by client applications is:

- Audio/Video In-Line Ads
  - Audio Content Types: MP3, AAC
  - Video Content Types: MPEG-4 (H.264), FLV
- Ad Title
- Impression URLs
- Media Files using progressive delivery
- IFrame and HTML Resources (see note below)
- Companion Banners
  - Multiple Sizes (see below for banner selection details)
  - Banner Content Types: JPG, PNG, GIF
  - Video Clicks on Companion Banners (both Click-Through and Click-Tracking URLs)
  - Tracking Events: `creativeView`
  - Companion Click-Throughs
  - Banner Alt Text

Note: Players should support **both** IFrame and HTML Resources, as it is supported by all On-Demand Ad sources. Players that are unable to support HTML and IFrame Resources should implement support for Static Resources, but this will limit the number of banners that can be displayed, as only Tap can return Static Resources.

Note: Players should support as much of the VAST specification as is practical (in addition to the minimum subset described above). As the On-Demand Ad Service evolves, more VAST features will eventually be supported (for example, Non-Linear Ads, or additional Tracking Events), which would require updates to client applications if they don't already support the full VAST 2.0 specification.

4.1.1 Empty Response

If the On-Demand Ad Service has no available ads that match the request's parameters (e.g., no ads are currently scheduled, or nothing has been found that matches the provided asset constraints or targeting profile), the On-Demand Ad Service will still return a `200 OK` reply, but with an empty VAST, such as:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VAST version="2.0.1"/>
```

4.1.2 JSONP Wrapper

If the rendering format requested is `vast-jsonp`, the VAST 2.0 response will be wrapped in a JSONP JavaScript fragment. This format requires an additional query parameter (`jscb`), which specifies the JavaScript function name that will be used in the JSONP response.

For example, if the Ad Request specifies the following parameters:

```
fmt=vast-jsonp&jscb=myJSFunctionName
```

The response will be as follows:

```
var vastXml="<content of VAST XML here>";
myJSFunctionName(vastXml);
```

The player is responsible for providing the implementation of the JavaScript function specified by the `jscb` parameter.

4.1.3 Selecting Companion Banners for Display

Players are responsible for selecting the appropriate companion banners for display based on, for example, the reserved areas and their sizes (i.e., a Web page containing placeholders for IAB Leaderboards and Big Boxes, or a mobile player with space for a phone-sized banner). Ads may have multiple companion banners of various sizes. When sending the ad (as VAST XML) to the player, it will include all of the ad's banners.

The player must iterate over the companion ads, selecting ads by size (i.e., width/height). It should select the ads for the sizes it is able to display, and ignore ads it is incapable of displaying. The player **should avoid** scaling the ad up or down, as this usually leads to bad image quality. In the event that the player *must* scale the image, the scaling must be applied such that it maintains the image’s original aspect ratio.

Every companion banner will offer three renderings: IFrame, HTML, or Static Resource. Depending on the player's implementation and runtime platform, one of these renderings should be more appropriate for display.

The player **MUST** call the various tracking URLs for selected ads. It **MUST NOT**, however, call the tracking URLs for companion banners that are not displayed. The player **MUST ONLY** call the tracking URLs for the rendering format used (e.g., the Static Resource's tracking URLs should not be called if the player uses the IFrame rendering), as every rendering already provides the elements necessary to perform its own tracking (e.g., transparent pixels embedded in IFrame/HTML renderings).

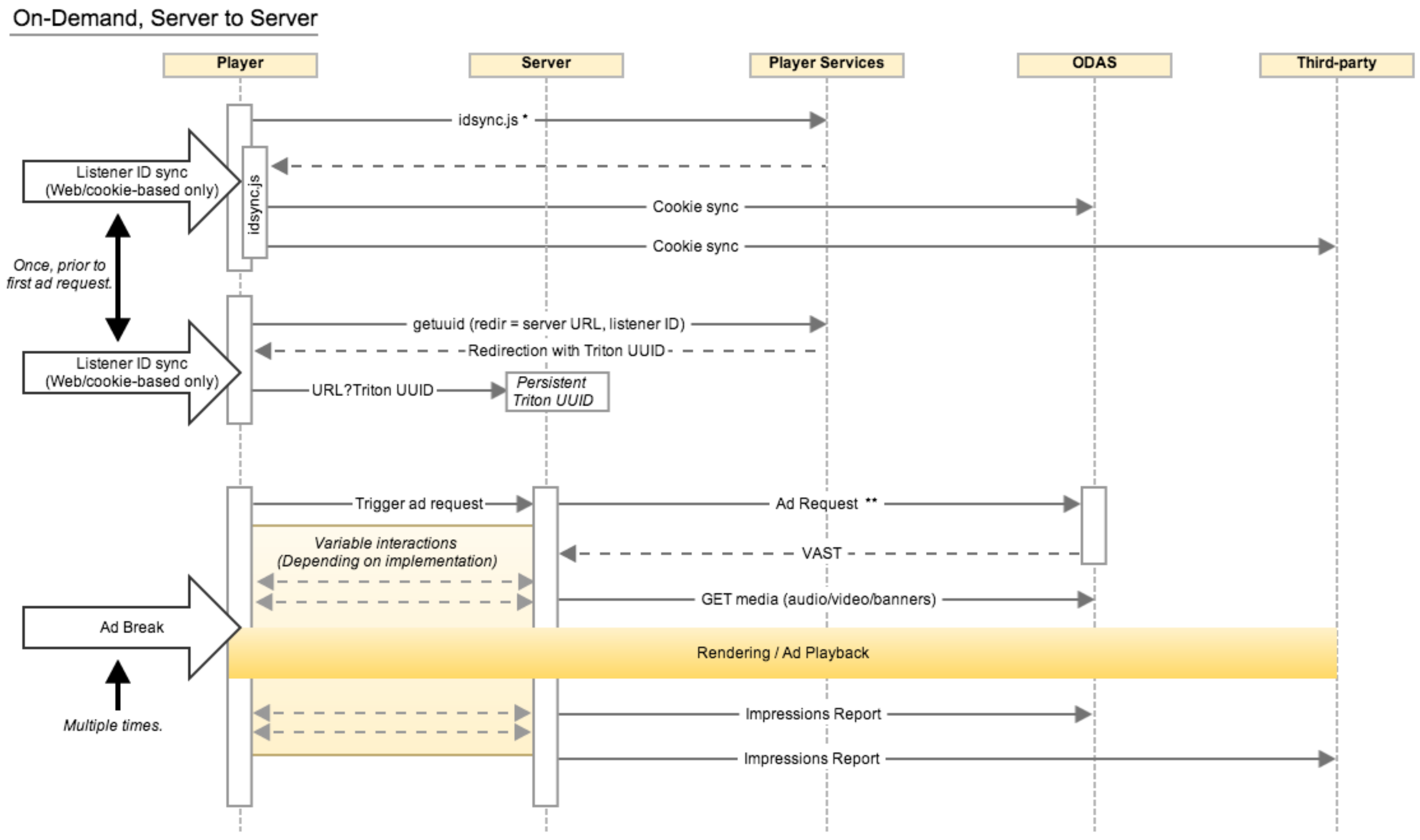
If an error occurs while retrieving a companion banner's rendering or resources (e.g., for URL-based resources, such as IFrame or Static Resource), and the player is thus unable to display the companion ad, the tracking URLs **MUST NOT** be called (as the ad was never actually displayed to the listener).

5. Server-to-Server Usage

Server side applications that make calls to the On-Demand Ad Service are responsible for collecting and passing all the client information that would affect ad selection to the On-Demand ad service. This includes:

- Listener ID
- Client's IP address, HTTP referrer, User Agent string.
- For web-based apps, the server application must ensure additional cookie synchronization functions are called from the client, as described in "Additional Parameters" on page 23.

5.1 Flow Diagram: Server to Server


On-Demand, Server to Server
Player
Server
Player Services
ODAS
Third-party
idsync.js *
idsync.js
Listener ID sync (Web/cookie-based only)
Cookie sync
Cookie sync
Once, prior to first ad request.
getuuid (redir = server URL, listener ID)
Redirection with Triton UUID
Listener ID sync (Web/cookie-based only)
URL?Triton UUID
Persistent Triton UUID
Trigger ad request
Ad Request **
Variable interactions (Depending on implementation)
VAST
GET media (audio/video/banners)
Ad Break
Rendering / Ad Playback
Multiple times.
Impressions Report
Impressions Report

* **idsync.js:** Contains all third-party ad servers and DMPs.

** **Ad Request:**
- Listener ID (Triton UUID)
- Targeting attributes
- Asset constraints
- Type (pre-roll/mid-roll)
- IP/Agent/Referrer
- No cookies

5.2 Additional Parameters

When doing server-to-server ad requests, the third-party server must add some additional query parameters to pass in the listener's information. These parameters must be specified in addition to the standard query parameters described in section 3.3, and are mandatory.

Parameter Name	Description	Comments
`lsid`	Listener ID	See "Listener ID Management" on page 7 and "Server-to-Server Cookie Synchronization" on page 23 for details.
`ip`	Listener's IP address	Dotted decimal format (e.g., `1.2.3.4`).
`ua`	Listener's User Agent	URL-encoded string.
`referrer`	Listener's HTTP Referrer	URL-encoded string. For web-based players, contains the URL to the web page the player is embedded in. For applications, this value should be empty.

5.3 Server-to-Server Cookie Synchronization

The player and the third-party server must use the same Listener ID for their On-Demand Ad Service requests (i.e., when requesting ads, reporting impressions, click-tracking, etc.). To achieve this, a cookie synchronization operation must be performed between the player and the server before ad requests can take place.

To do this, the web page that embeds the player must add a call to the following service:

```
http://playerservices.live.streamtheworld.com/api/getuuid?redir=<redirection-
url>
```

The `<redirection-url>` in the URL above is a URL to a service on the third-party's side. The value of the `redir` parameter must be URL-encoded. The URL should include a `@UUID@` token, which will be replaced by the Triton Listener ID before performing the redirection. When this service is called, the player's Triton Listener ID is extracted from its cookies (or a new Listener ID is generated and set), token replacement occurs, and the player is redirected to the target URL.

For example:

```
http://playerservices.live.streamtheworld.com/api/getuuid?redir=http%3A%2F%2Fhos
t.domain.tld%2Fpath%2Fservice%3Fid%3D@UUID@
```

This would redirect the player to:

```
http://host.domain.tld/path/service?id=49a9b47f-298a-4c4d-b07a-3da46909cf63
```

On the third-party server side, the redirection URL's target handler must store the Listener ID received (in its database, or in another cookie, for example), and send that Listener ID along with any On-Demand Ad Service server-to-server requests (via the `lsid` parameter).

Note: As the easiest and most reliable way to add this URL is usually by using an `<img src="..."/>` HTML tag, it is suggested that the redirection target URL return a 1x1 transparent GIF pixel.

6. Privacy Policy Considerations

You should review your privacy policy and terms of service to ensure they accurately reflect the targeting considerations enabled by on-demand advertising and comply with applicable laws and regulations, and revise them if necessary. Specifically, you need to ensure your player and/or your website contains a privacy policy notification that:

(i) discloses the usage of third-party tracking technology including, without limitation, cookies, web beacons, clear .gifs or similar technologies;

(ii) discloses the collection and usage non-personally identifiable information (Non-PII) by third-parties for the purpose of delivering advertising content to the End User based on the End User's online behavior and past browsing activities or certain targetable attributes;

(iii) contains a conspicuous live hyperlink to an opt-out web site that provides the End-User the ability to opt out of interest-based advertising through the Services; and

(iv) discloses or contains any additional information required by applicable laws and industry practices.

Please update your privacy policy accordingly.

To include this disclosure information you can either:

- Update your privacy policy on your website and include all the above mentioned items; or
- Include a privacy link on your player that points to the Triton Digital streaming platform privacy policy: http://tritondigital.com/privacy-policy-advertising-platform

7. Appendix: Using a2x

7.1 a2x and On-Demand Advertising FAQ

Use the following questions and answers as a guide to implementing a2x in your on-demand environment.

Q: Can I control the time length (duration) of ads that come from a2x?

Yes, but with limitations. By default, your a2x account is set up to allow ads of the following durations:

- 15 seconds
- 30 seconds
- Other (e.g., 45 seconds, 60 seconds, etc.)

You cannot restrict the duration within the ad request, but if you want to limit the ads you receive to fewer options (e.g., only allow 30-second ads, or allow 15- and 30-second ads, but not "Other" durations), contact Triton Digital and we will configure that restriction into your a2x account setup.

However, be aware that the success of this restriction depends on how the advertisers tag their a2x ads. For example, if you restrict your durations to **only 30-second ads**, and an advertiser tags their 30-second ads as "other," you will not receive them. Also, if an advertiser mistakenly tags a 15 second ad as 30 seconds, you will receive that ad (assuming it meets your other targeting criteria).

Note that Triton Digital makes every effort to ensure that advertisers respect the ad duration settings and tag their ads properly. Also, **we recommend you do not configure any duration restrictions**; using the default settings ensures the best performance and provides the highest degree of demand and ad revenue.

Q: What if my app cannot display the companion (sync) banners provided with an a2x ad?

When configured properly, a2x only tracks audio impressions. So if a companion (sync) banner returned by a2x cannot be displayed by your app, it is simply ignored. The audio plays, and the impression is tracked on the audio asset. However, we strongly suggest you configure your player/app to enable as many banner sizes as possible, paying particular attention to the most common ones. This will help ensure a better user experience and engagement from your listeners (including click-throughs) and better value from your a2x ads.

Q: Why am I not seeing a2x ads in my test environment?

Most a2x ads are delivered to listeners who have been targeted based on their cookie or device ID profiles. In a micro environment with only a handful of users – such as some test environments – you will likely not see any a2x ads returned by the ad server because of the lack of targeting matches. On a macro level, with thousands of ad requests, you will see a2x ads appear.

If you are not seeing a2x ads, try using the accounts described in "Test Accounts for a2x," below, to complete your testing. They deliver non-targeted a2x ad campaigns.

7.2 Test Accounts for a2x

The accounts described below contain sample ad campaigns set to deliver **audio only** and **audio with sync banners**.

a2x Only

a2xONLYTestBED, Station ID 77813, ONLY RETURNS ONE a2x ad:
http://cmod208.live.streamtheworld.com/ondemand/ars?type=midroll&fmt=vast&stid=77813

Tap Server-to-Server Mode

ODTESTBED, Station ID 23193, standard client app to server ad requests:
http://cmod209.live.streamtheworld.com/ondemand/ars?type=midroll&stid=23193

Tap Only (No Delegated Ads, No a2x)

ODTESTBED S2S Station ID 76383, for apps using server-to-server method (see specs):
http://cmod209.live.streamtheworld.com/ondemand/ars?type=midroll&stid=76383

7.3 Sample a2x Ad (VAST File)

```
<VASTversion= "2.0.1">
  <Adid= "a2x-12947352-782916996284056734">
    <InLine>
      <AdSystemversion="3.0">
        <![CDATA[ a2x]]></AdSystem >
        <AdTitle><![CDATA[N/A ]]></AdTitle>
        <Impressionid="ImpressionConfirmationURL">
           <![CDATA[http://cmod208.live.streamtheworld.com/ondemand/impression? cat=a2x-
ondemand-midroll&cid=a2x-12947352- 782916996284056734&stid=77813&lsid=cookie:d45a0f70-a04b-
4efa-9d94- dc2c1ae112d2&cb=709776561&sourceInfo=buyer_id%3D1577%26cpm_paid%3D8.0%26asset_typ
e%3Daudio%26brand_id%3D28222%26injector_type%3Dondemand%26cpm_bid%3D8.0%26ad_type %3Dmidroll
]]>
        </Impression>
        <Creatives>
          <Creativesequence="1">
            <Linear>
              <Duration>00:00:30.000 </Duration>
              <MediaFiles>
                <MediaFiledelivery="progressive" type="audio/mpeg" width="0" height= "0">
                  <![CDATA[http://cmodmedia208.live.streamtheworld.com/media/appnexus-
audio/a2x-02aca3b663a4c2769f20354b706ec7f7/8a319fc8026260e1b0e6396c1bc049fc_5039b9f21152c5b
b06a8b5c5fba67097.mp3]]>
                </MediaFile>
              </MediaFiles>
            </Linear>
          </Creative>
          <Creativesequence="1">
            <CompanionAds>
              <Companionwidth="300" height="250">
                <IFrameResource><!
[CDATA[http://cmod208.live.streamtheworld.com/ondemand/viewtrack?cat=a2x-ondemand-
midroll&cid=a2x-12947352-782916996284056734- bigbox&stid=77813&lsid=cookie:d45a0f70-a04b-
4efa-9d94- dc2c1ae112d2&cb=709776561&sourceInfo=buyer_id%3D1577%26cpm_paid%3D8.0%26asset_typ
e%3Daudio%26brand_id%3D28222%26injector_type%3Dondemand%26cpm_bid%3D8.0%26ad_type
%3Dmidroll&redir=http%3A%2F%2Fib.adnxs.com%2Ftt%3Fid%3D2317689]]>
                </IFrameResource>
             </Companion>
           </CompanionAds>
          </Creative>
       </Creatives>
      </InLine>
     </Ad>
   </VAST>
```