Video Ad Serving Template (VAST)

VERSION 4.0

Released January 21, 2016

This document has been developed by the IAB Digital Video Committee

The Video Ad Serving Template (VAST) specification was created by a working group of volunteers from IAB member companies.

The following IAB member companies contributed to this document:

4A's & Ad-ID
A&E Television Networks
ABC/Disney Interactive Media Group
Adobe Systems Inc.
AerServ
AT&T
Brightcove
Brightline
Brightroll
CBS Interactive
Collective
Demand Media
DoubleVerify
Engage BDR
Extreme Reach
EyeSee
Fox Networks Group
FreeWheel
Google & YouTube
Hulu
Innovid Inc
Integral Ad Science
JW Player
LiveRail Inc.
Mediative
Microsoft
Mixpo
Moat
NBC Universal
Opera Media Works
PGA Tour
PointRoll
Pubmatic
Quantcast
Rhythm New Media
Rovi Corp
SeaChange International
SET
Simulmedia
Sizmek Technologies
Sony
SpotX
Time Warner Cable
Tremor Video
TubeMogul
Turner Broadcasting System, Inc./CNN.com
Twitch
Vdopia
Verizon Wireless
Vevo
Viacom
Vindico
Visible Measures
VivaKi
White Ops
Xumo

The IAB leads on this initiative were Jessica Anderson and Luke Luckett.

Contact adtechnology@iab.com to comment on this document. Please be sure to include the version number of this document (found on the bottom right corner on this page).

About the IAB's Digital Video Committee

The Digital Video Committee of the IAB is comprised of over 180 member companies actively engaged in the creation and execution of digital video advertising. One of the goals of the committee is to implement a comprehensive set of guidelines, measurement, and creative options for interactive video advertising. The Committee works to educate marketers and agencies on the strength of digital video as a marketing vehicle. A full list of Committee member companies can be found here.

This document is on the IAB website at: http://www.iab.com/guidelines/vast4.0

Table of Contents

Executive Summary

VAST is a Video Ad Serving Template for structuring ad tags that serve ads to video players. Using an XML schema, VAST transfers important metadata about an ad from the ad server to a video player. Initially launched in 2008 VAST has since played an important role in the growth of the digital video marketplace.

The early days of video consisted mostly of shared videos and other user-generated content. Success in monetizing this content with ads has produced the resources to improve the digital video marketplace. However, digital video has met a number of challenges along the way.

One challenge, and a key reason some video publishers avoid using VAST, is a lack of quality control. Along with the IAB Video Player-Ad Interface Definition (VPAID), VAST can deliver ads programmatically or include ads with complex interactions. If a player isn't programmed to accept VPAID ads, the ad cannot be executed. Even when the player does accept VPAID ads, performance may be slow and cause latency in load times. In the meantime, the audience experiences a delay or a malfunction in their viewing experience.

Publishers and ad vendors need a way to separate the video file from its interactive components to ensure that ads play in systems that cannot execute the interactive components. These ads should also execute more efficiently in players that are equipped to handle the interactions.

Another challenge, especially for broadcasters who are moving their content online, is the lack of a consistent identifier for creative that is maintained across systems. VAST offers a creative identifier, but it has been used inconsistently and one creative may use different identifiers for every system it passes through. A system-wide identifier is a requirement for broadcasters trying to maintain control over the ads they play.

VAST 4.0 has addressed these challenges along with a few others. As players begin to adopt the updates in VAST 4.0, digital video can expect to see smoother operation and the continued growth that results.

The updates made in VAST 4.0 and the challenges they address are summarized here:

- **Separate video file and interactive file:** The complexity of digital video has given rise to the need to separate the linear video file from any creative interactive API files. While the VAST media file has accepted a variety of media files in the past, interactive APIs cannot always be executed. A VAST tag that provides the video file separate from APIs can display more successfully across platforms and devices.
- **Server-side support:** While client-side ad execution and tracking has been the recommended way to track ad impressions and other metrics, digital in-stream video ads are often served to devices (clients) that cannot execute and track ads using traditional display methods. VAST 4.0 supports the increasingly common “ad-stitching” method for stitching linear video ads into a video content stream and sending it to players with limited capabilities.
- **Mezzanine file:** To support advertising across video platforms that include long-form content and high-resolution screens, VAST 4.0 features include support for the raw, high-quality mezzanine file. The mezzanine file is very large and cannot be used for

ad display, but ad-stitching services and other ad vendor use it to generate files at appropriate quality levels for the environment in which they play.

- **Ready-to-serve files:** Along with support for including the mezzanine file, VAST 4.0 provides guidance on providing three ready-to-serve video files, each at different quality levels, to ensure that a linear video ad can always play. The IAB Digital Video Ad Format Guidelines offers guidance on video file specifications for linear ads.
- **Universal Ad ID:** While VAST has offered a creative identifier in the past, it has been used inconsistently. The new Universal Ad ID feature is used specifically for including a creative identifier that is maintained across systems. The existing `adId` attribute for creative can still be used to log creative IDs specific to the server.
- **Ad Verification and Viewability Execution:** Verification vendors have been using VPAID for measurement verification instead of using it for ad interaction as VPAID was intended. VAST 4.0 offers a designated space for inserting ad verification APIs, enabling a more streamlined process for executing files strictly intended for ad verification. In addition, a secondary impression element, the `<ViewableImpression>` element, has been added to allow publishers the option to track viewability on their inventory.
- **Support for categories:** Ad categories help video publishers separate competing ad creative and improve brand safety. VAST 4.0 ad categories support these efforts.
- **Conditional ad declaration:** In programmatic environments, a VPAID unit is sometimes used to decide whether or not to place an ad. If this “conditional ad” never results in an ad to display, the publisher may have to forfeit any revenue from the resulting lost inventory. A declaration in VAST for a conditional ad helps publishers prevent and reclaim any potentially lost inventory revenue in programmatic ad delivery.
- **New error codes:** Along with support for the mezzanine file and other new features, added error codes provide additional troubleshooting support.
- **Standardized timestamp:** Trackers used in VAST often include timestamp macros, but its use has not been consistent. In VAST 4.0, the [TIMESTAMP] macro and the format for time has been standardized to enable more consistent time-sensitive tracking.

Intended Audience

This document was designed for digital video technologists who either develop players that accept digital in-stream ads or for vendors who develop ads to be sent to digital in-stream players.

For engineers, section 3 defines all the VAST XML elements. Section 5 includes a “human-readable” schema for quick reference with links to more details in the document if needed. Section 2 defines VAST compliance and section 4 provides technical tips for migrating to VAST 4.0.

For video executives, the executive summary and section 1 provide high-level explanation of how VAST can be used to streamline digital video ad operations.

Resources for Digital In-stream Video

In order to improve the interconnectivity of the digital video marketplace, the IAB has published technical specifications, metric definitions, and best practices developed by members with industry experience. Descriptions for each of these publications are listed below.

- **VAST:** (this document) the Video Ad Serving Template is an XML response framework that enables a consistent delivery format for ad across streaming video platforms.
- **VPAID:** the Video Player-Ad Interface Definition specifies the protocol between the ad and the video player required to enable ad interactivity and other advance video advertising functionality
- **VMAP:** the Video Multi-Ad Playlist is an XML response framework that defines where to place ads within the video content.
- **Digital Video Ad Metric Definitions:** an industry-defined list of metrics used in linear and nonlinear in-stream video ads.
- **Digital Video Ad Format Guidelines:** an industry-defined list of streaming video creative submission specifications.
- **DAA Interest-Based Advertising (IBA) notice for digital video:** Guidelines for implementing the AdChoices program within in-stream ads that are placed using interest-based criteria.

1 General Overview

VAST is used to send in-stream ad details to a video player. Historically, the player (client) has received, executed, and tracked streaming video ads. However, with the increase in player devices, the player is often unable to execute anything more than a single stream of content. Players might have compensated for this by using one player for content and loading a secondary player for ad playback. After ad playback, the original player would be reloaded for resuming content playback. This process caused a brief buffering period between player loads.

The solution that has emerged for this challenge is a service that involves inserting ads into a stream of content for the player. The result is a seamless experience for the viewer along with the ability to select ads dynamically for insertion and more sophisticated tracking options.

VAST 4.0 includes support for high-quality video formats necessary for long-form video content and server-side tracking for use when ad-stitching is leveraged to reach devices that cannot use client-side tracking methods. Version 4.0 also allows embedding optional scripts for viewability and ad verification. Further details for using VAST with VPAID can be found in additional documents on the IAB website.

1.1 VAST Ad Serving and Tracking

Display advertising uses standardized browser technology to request and execute ads. However, digital in-stream video advertising operates on players, sometimes built with proprietary code. As a template for ads served to a video player, VAST offers a set of instructions for developers on how to program their players to process VAST-formatted ads. Using VAST, ad servers can serve ads to any VAST-compliant player regardless of what code the player uses.

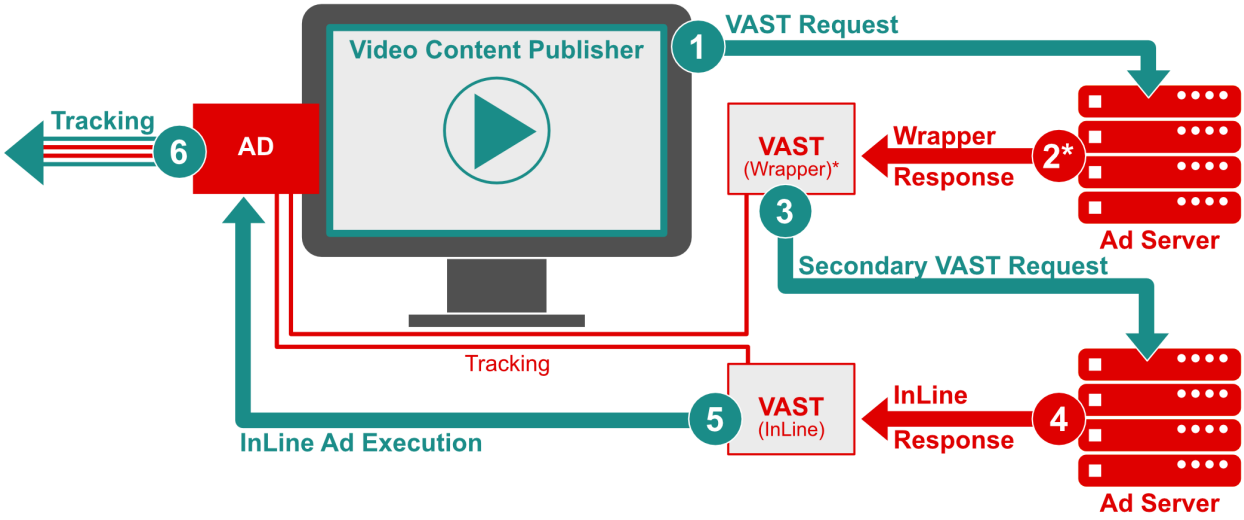
1.1.1 Sending An Ad Request

VAST provides a template for serving an ad to a video player. While VAST does not provide ad request details, VAST does expect the player to declare what ad types it supports (Section 2.3.1). This declaration can be made as part of its VAST compliance status, but as automated processes are used more in video, the player needs a programmatic way to identify what type of ad it needs.

The IAB Tech Lab is working on a solution for consistency in making ad requests. To find out more or to get involved, email adtechnology@iab.com.

1.1.2 Client-Side Ad Serving

VAST is a unidirectional means of sending ad details to a video player. Built as a layer on top of browser technology, the VAST process that uses client-side execution looks something like this:



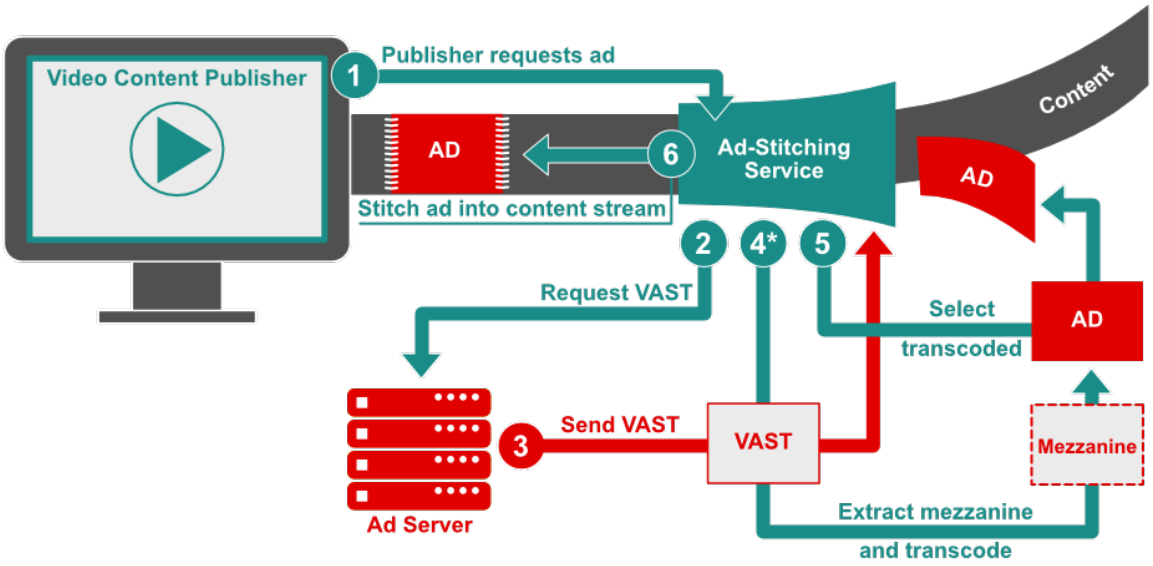
VAST Request: At some point during video content playback, either before (pre-roll), in the middle of (mid-roll), or after (post-roll), the player reaches a cue to insert an ad and uses HTTP to send the request for an ad. See section 1.1.1 on sending an ad request. The request is sent to the primary ad server, which may be the video publisher's ad server or a supply-side platform (SSP).

1. *** Wrapper Response:** The primary server responds with VAST. This response is either an InLine response or a Wrapper response. If the server can fill the ad request, it sends an InLine response (step 4). In many cases, the ad server redirects the player to a secondary server using a Wrapper response.
2. **Secondary VAST Request:** If a wrapper response is received, the player make a secondary request to another server. The secondary response may be an InLine response or another VAST Wrapper.
3. **InLine Response:** Eventually, after a series of requests and responses, an ad server provides an InLine response.
4. **InLine Execution:** The video player executes the VAST response.
5. **Tracking**: At key points during ad playback, tracking information is sent for both the InLine and Wrapper responses that the player received. In traditional client-side ad serving, cookies are used to track ads and the computers on which they play.

1.1.3 Server-Side Ad Serving

The example just described the general process for serving an ad directly to a video player, the client, and uses client-side tracking. With client-side tracking, the player sends tracking information. However, in today's wide array of streaming video players the player may not be capable of executing dynamic ad responses or tracking impressions and interactions. In these cases, an intermediary server is needed to insert ads dynamically into the video stream.

Called ad stitching among other terms (stream stitching, ad insertion, etc.), the process looks something like this:



1. **Publisher requests ad:** The video publisher sends an ad request to the ad-stitching service.
2. **Request VAST:** The ad-stitching services makes a request to the ad server for a VAST tag.
3. **Send VAST:** The ad server sends a VAST tag with a mezzanine file and ready-to-serve files. If the ad stitching service has already received the creative for a previous request and has transcoded the mezzanine file, then it moves on to step 5.
4. ***Extract mezzanine and transcode:** The ad-stitching service pulls the unique creative identifier from the VAST tag. If the creative has never been used in the system, the mezzanine file is extracted and transcoded. In this scenario, the ad is skipped and the next available ad is played instead. VAST error code 407 is sent.
5. **Select transcoded:** If the creative in the VAST tag from step 3 matches the unique creative identifier for an ad that has already been transcoded, the ad-stitching service selects the pre-transcoded file already in the system.
6. **Stitch ad into content stream:** The ad-stitching service stiches the ad into the content stream and serves the content and ad to the player in one continuous stream.

Ad-stitching vendors rely on a unique creative identifier for managing the mezzanine source file and its cache of transcoded files for stitching into a video stream. If the ad creative is changed in any way, it should be served with a new creative identifier. In VAST 4.0, the unique creative identifier is provided in the `<UniversalAdId>` element under `<Creative>`. See section 3.7.1 for details.

1.1.4 Server-to-Server Ad Tracking

With client-side ad tracking described in section 1.1.2, the player (client) sends tracking included in the VAST tag and uses cookies to determine which computers executed the ad. However, in server-to-server and server-side ad-stitching, the player may not be able to process ad tracking, and the ad-stitching service cannot access cookies used in traditional client-side tracking. Instead, the ad-stitching service must identify devices where ads play by a combination of other methods.

When an ad-stitching service is involved, the ad-stitching server may send tracking on the player's behalf. This server-to-server tracking process is problematic because all the tracking is coming from one IP address. To an ad server that is receiving tracking information, the reports look similar to faked traffic fraud.

Ad stitching service providers can avoid being mistaken for fraud by including headers that identify the client IP address and user agent string as follows:

> **X-Forwarded-For:** a header that identifies the client IP address. This value should match the value, if included, in the IP field of the device object as part of the ad request.
>
> **X-Device-User-Agent:** a header that identifies the client's device user agent string. This value should match the value, if included, in the UA field of the device object as part of the ad request.

A secondary option for identifying the originating IP address and user agent is the Forwarded HTTP Extension, which can disclose all proxy information in one field, making it easier to correlate them. While this option may be more efficient, more systems are primed to look for the X-Forwarded-For and X-Device-User-Agent fields. These more commonly used header fields should be used first, and optionally, the Forwarded HTTP Extension may also be included.

While these recommendations for tracking support server-side tracking, IAB Impression Measurement Guidelines developed with the MRC favor client-side tracking. No guidelines currently exist for server-side impression tracking. However, the key ideal behind IAB impression counting is to count the impression as close as possible to the opportunity to be viewed. In video, the ad is ideally counted after the linear video ad has buffered.

In general, an ad-stitching service may have little or no control over ad play once the ad is stitched into the content and streamed to the client. Impression reporting may vary by implementation. For the ad stitching service in situations where the client cannot count impressions, an impression could be reported as the ad is sent on the stitched stream and therefore be as close as possible to the opportunity to view. Alternately, a session-oriented ad-stitching service may report impressions from a given session at session completion. However, any impression measurement beyond the ad-stitched stream is out of the ad-stitching services' control and should be counted by the player whenever possible.

Auditing for compliance with IAB Impression Measurement Guidelines should focus on disclosing the process by which impressions are counted and any limitations with reporting impressions in certain situations and environments.

See section 2.3.4 for more information on tracking.

1.2 Viewability and Ad Verification

VPAID, the Video Player-Ad Interface Definition, is designed to offer bilateral (two-way) communication between the ad and the video player. Some ad verification services have adopted VPAID to validate ad playback duration, position within browser and other qualities designed to verify that an ad had an opportunity to be viewed (viewability).

When using VPAID for these services, several data-collection wrappers may be applied. These wrappers can delay page rendering and create a negative experience for the viewer.

To support viewability and verification efforts without sacrificing performance, VAST 4.0 offers an `<AdVerification>` element where vendors can place their code.

The `<AdVerification>` element supports a JavaScript resource and a Flash resource. This script resource enables the player to provide requested details about ad interaction and playback. Unidirectional scripting from the player to the ad is allowed, and the player may choose whether it will execute the script.

Any VPAID-based code used for ad verification should be placed under the `<AdVerifications>` element described in section 3.16. Alternately, any ad verification code can be placed under this element. Regardless of code provided (VPAID or proprietary), the player must be equipped to execute the code. For example, if VPAID is provided for the purpose of ad verification, the player must be VPAID-enabled in order to respond to the verification requests.

As part of the VAST 4.0 initiative, a proposal has been submitted to the IAB TechLab to develop a solution for consistent ad verification data collection while minimizing the impact on performance.

For information on the status of this project or to get involved, email adtechnology@iab.com.

See section 3.16 for details.

1.3 Programmatic Support

While VPAID was never intended for any other purpose than to provide a channel for player-ad interaction, VPAID is nevertheless being used to support video ad serving in an increasingly complex marketplace. Some vendors have used VPAID as a placeholder while the system decides which ad it want to serve. Served as a VAST InLine ad, the player expects and ad. However, since the server is only looking for an ad to serve instead of providing the ad as expected, the player experiences a timeout while it waits. In some cases an ad is never served. When this happens, the publisher forfeits revenue reserved for that placement.

Video publishers need a way to know ahead of time whether the VPAID unit provides a direct link to an ad or used as a placeholder while a suitable ad is found (conditional ad). This conditional ad notification needs to be included in the VAST response. If the notification is included in the VPAID framework, it comes too late in the delivery process. As part of the response framework, the early notification gives the player time to decide if it will accept a conditional ad or move on.

In VAST 4.0, an attribute for `conditionalAd` has been included within the `<Ad>` element. Using VAST to identify a VPAID unit that might not contain an ad helps video publishers fill available inventory when buyers back out. The improved transparency also improves the viewer experience.

1.4 Long-Form Video Support

Long-form video is the extension of traditional broadcast networks to digital mediums. To enable video advertising across screens that include TV content, the response framework in VAST needs to reduce the challenges faced in this digital video environment. Specifically, VAST needs to support the following features:

- Server-to-server ad stitching
- Availability of the high-quality source (mezzanine) file for the ad
- A unique identifier that follows creative and related data from system to system

1.4.1 High-Quality Video

Previous versions of VAST have allowed for multiple media files so that the video player can poll for the file best suited to the environment where the ad will play. However, the high standards for quality in long-form video need more than a few options.

VAST 4.0 includes a media container for the mezzanine file, which is the raw high-quality video file that the publisher can use to produce the best quality file where needed. In addition to the mezzanine file, VAST 4.0 requires either an adaptive stream ready-to-serve file or a minimum of three media files at different levels of quality: high, medium, and low. Identifying the quality levels of three media files enables the video player to more quickly find the appropriate file needed for a given environment. A separate document, the IAB Digital Video Ad Format Guidelines, is provided for ad developers and outlines encoding recommendations for each of these files.

See section 3.9 for details.

1.4.2 Unique Creative Identification

As ad creative move from system to system, they become more difficult to track and impressions involving those creative become difficult to reconcile in reports from different systems. Historically, VAST has provided a placeholder for a creative id, but its purpose has been unclear and its use varies under different vendors and use cases. In long-form video, the need for unique creative identification that spans multiple systems has become vital to extending digital video across platforms, especially where long-form content is viewed.

In VAST 4.0, the placeholder for a unique creative ID has been pulled out into a new element to draw more attention it and provide attributes that more clearly define the id. The new `<UniversalAdId>` element is required for linear ads in long-form video and offers two attributes: one for defining the `idRegistry` and one for the `idValue`.

Using a unique creative identifier enables all data associated with the creative to follow across systems. Unifying data under a unique ID, streamlines data collection operations, reporting, and analysis.

See section 3.7.1 for details.

2 VAST Compliance

Compliance is a two-party effort that involves, at a minimum, the video player and the ad server. Both must meet certain expectations so that VAST can be truly interoperable and encourage growth in the marketplace.

General Implementation Note	IAB VPAID and VMAP specs are excluded from VAST compliance because these specs are independent of each other and of VAST. Compliance with one spec does not imply compliance with any of the other specs. Compliance for either spec must be separately declared.

2.1 Ad Server Expectations

VAST-compliant ad servers must be able to serve ad responses that conform to the VAST XML schema defined in this document. Ad servers must also be able to receive the subsequent tracking and error requests that result from the video player's execution of the VAST ad response.

2.2 Video Player Expectations

VAST-compliant video players must be able to display the ad in a VAST response according to the instructions provided by the VAST ad response and according the video player's declared format support, which includes:

- Rendering the ad asset(s) correctly
- Respecting ad server instructions in a VAST response including those of any subsequent ad servers called in a chain of VAST wrapper responses, providing the responses are VAST-compliant
- Responding to supported user-interactions
- Sending appropriate tracking information back to the ad server
- Supporting XML conventions such as standard comment syntax (i.e. acknowledge VAST comments in the standard XML syntax: `<!--comment-->`)

Details for proper ad display and VAST support are defined throughout this document.

2.3 General Compliance

VAST specifies both the format of the ad response and how the video player should handle the response. In order for VAST to be effective, both ad servers and video players must adopt the guidelines outlined in this document.

In general, the video player need only accept ads that it requests and ad server responses should be displayed in the ad format intended.

For example, VAST allows for compliance while only supporting a subset of ad types (described in section 2.3.1). If a video player requests a Nonlinear Ad but receives a Linear Ad, the video player is not expected to display the Linear Ad. Similarly, if a standard Linear Ad is requested but a Skippable Linear Ad is received, the video player is not expected to

display the Skippable Linear Ad nor should the video player play the Skippable Ad as a Linear Ad (without skip controls).

The following features must be supported for general functionality:

- Declaration of ad types
- XML structure
- Tracking
- Wrappers
- Error reporting
- Macros
- Industry icons

These features are described in the following sections.

2.3.1 VAST Ad Types

VAST covers several distinct video ad types, but ad serving and video publisher organizations may not want to support all formats. For example, some vendors may choose to serve only linear ads with companions. Likewise, some publishers may only want to support nonlinear ads in their video players.

VAST 3.0 introduced five video ad types for compliance so that organizations may be compliant with VAST while only supporting a selected subset of the ad types.

The VAST compliance ad types are as follows:

1. Linear Ads
2. Nonlinear Ads
3. Companion Ads
4. Skippable Linear Ads
5. Ad Pods

A company wishing to display IAB's seal for VAST compliance must declare which of the five ad types their technology supports.

2.3.2 XML Structure

A VAST-compliant ad response is a well-formed XML document, compliant with XML 1.0 so that standard XML requirements such as character entities and `<!--XML comments-->` should be honored. It must also pass a schema check against the VAST 4.0 XML Schema Definition (XSD) that is distributed in conjunction with this document.

Ad Server Implementation Note	All URIs or any other free text fields containing potentially dangerous characters contained in the VAST document should be wrapped in CDATA blocks. The VAST response should be carefully tested for appropriate treatment of URI characters that require special handling.

2.3.3 Encoding URIs for VAST

URIs provided in a VAST response must be CDATA-wrapped as in the following example:

```
<Impression id="myserver">
      <![CDATA[
      http://ad.server.com/impression/dot.gif
      ]]>
</Impression>
```

Wrapping the URI in a CDATA section enables most characters to be included as they are. For example, without a CDATA section, the character `&` would need to be encoded as `&`. However, encoding this within a CDATA section double-encodes the URI.

Consider the CDATA wrapping needed for the following URI:

```
http://ad.server.com/impression/dot.gif?v=1&id=abc
```

```
<Impression id="myserver">
      <![CDATA[
      http://ad.server.com/impression/dot.gif?v=1&amp;id=abc
      ]]>
</Impression>
```

The encoding of `&` into `&` is not necessary in this example because the URI is enclosed in a CDATA section. Characters in the URI that are also used to section out the URI with CDATA may need extra encoding.

Consider the CDATA wrapping needed for the following URI:

```
http://ad.server.com/impression/dot.gif?s=x]]>x
```

```
<Impression id="myserver">
      <![CDATA[http://ad.server.com/impression/dot.gif?s=x]]
      ]]>
      <![CDATA[>]]>
      x
</Impression>
```

The `]]>` characters are used to close the CDATA section; therefore, the `>` character must be enclosed in a secondary CDATA section. Since the `x` is harmless character at the end, it can be left outside the CDATA section and will be concatenated with the other two URI components, each closed in their CDATA sections.

Since CDATA-wrapping URIs is a requirement in VAST, the author of the VAST response should carefully edit and test all included URIs, especially when input values require special handling. Incorrect treatment of these characters may cause ad playback to fail or enable content injection attacks.

Some addition examples are offered in the following table.

Impression URL: `http://ad.server.com/impression/dot.gif`

```
<Impressionid="myserver">
      http://ad.server.com/impression/dot.gif
</Impression>
```

Not enclosed in a CDATA section

```
<Impression id="myserver">
      <![CDATA[http://ad.server.com/impression/dot.gif]]>
</Impression>
```

Correct and backwards compatible

Impression URL: `http://ad.server.com/impression/dot.gif?v=1&id=abc`

```
<Impression id="myserver">
      http://ad.server.com/impression/dot.gif?v=1&amp;id=abc
      </Impression>
```

The `&` is xml-encoded, but the URI needs to be wrapped in a CDATA block

---

```
<Impression id="myserver">
      http://ad.server.com/impression/dot.gif?v=1&id=abc
</Impression>
```

Not only is this URI not CDATA enclosed but the `&` is also not xml-encoded

---

```
<Impression id="myserver">
      <![CDATA[http://ad.server.com/impression/dot.gif?v=1&amp;id=abc]]>
</Impression>
```

This URI is both CDATA-enclosed and the & is xml-encoded. The player will interpret the URI as:

```
http://ad.server.com/impression/dot.gif?v=1&amp;id=1234
```

```
<Impression id="myserver">
      <![CDATA[http://ad.server.com/impression/dot.gif?v=1&id=abc]]>
</Impression>
```

This URI is properly wrapped in a CDATA block. The `&` doesn't need to be encoded.

Impression URL: `http://ad.server.com/impression/dot.gif?s=x]]>x`

```
<Impression id="myserver">
      http://ad.server.com/impression/dot.gif?s=x]]&gt;x
</Impression>
```

Not enclosed in a CDATA section even though `>` is encoded

```
<Impression id="myserver">
      http://ad.server.com/impression/dot.gif?s=x]]>x
</Impression>
```

Not enclosed in a CDATA section

```
<Impression id="myserver">
      <![CDATA[http://ad.server.com/impression/dot.gif?s=x]]>x]]>
</Impression>
```

CDATA section used but appears to end early because of the `]]>` characters before the `x`, potentially allowing content injection attacks

---

```
<Impression id="myserver">
      <![CDATA[http://ad.server.com/impression/dot.gif?s=x]]]]>
      <![CDATA[>]]>x
</Impression>
```

Correct and backwards compatible

Lastly, a VAST-compliant ad response must conform to certain additional dependencies that cannot be expressed in the VAST 4.0 XSD. For example, one ad type of either `<InLine>` or `<Wrapper>` is allowed but not both. Another example is the protocol for providing 3 ready-to-serve media files, ideally separate from any interactive components (see section 3.9). The XSD can only validate for whether you've provided a URI under `<MediaFile>`; it cannot validate whether the appropriate files have been provided. Such dependencies are further described throughout this document.

2.3.4 Tracking

Tracking an ad served in VAST format is done using a collection of VAST tracking elements at different levels in the VAST response. These tracking elements each contain a URI to a resource file or location on a server defined by the ad server that sent the VAST response. The resource file is usually (but not always) a 1x1 transparent pixel image (i.e. tracking pixel) that when called, records an event that is specific to that tracking pixel.

Video Player Implementation Note	The video player is responsible for requesting tracking pixels at appropriate times during the execution of a VAST ad response.

Most tracking elements are optional for the ad server to include, but if included, the video player is required to request the resource file from the URI provided at the appropriate times, or "fire" the tracking element. Advertisers and publishers depend on accurate tracking records for billing, campaign effectiveness, market analysis, and other important business

intelligence and accounting. Good tracking practices throughout the industry are important to the success and growth of digital video advertising.

General Implementation Note	The video player must send requests to the URIs provided in tracking elements; however, the video player is not required to do anything with the response that is returned. The response is only to acknowledge an event and to comply with the HTTP protocol. This response is typically a 200 with a 1x1 pixel image in the response body (although the response could be of any other type).

The use of multiple impression URIs enables the ad server to share impression-tracking information with other ad serving systems, such as a vendor or partner ad server employed by the advertiser. When multiple impression elements are included in a VAST response, the video player is required to request all impressions at the same time or as close as possible to the same time. Any significant delay between impression requests may result in count discrepancies between ad serving systems.

Video Player Implementation Note	If multiple `<Impression>` elements are provided, they must be requested at the same moment in time or as close in time as possible. In particular for a VAST response containing a `<Linear>` element, compliancy with the IAB Digital Video Measurement Guidelines. If any of the requests are delayed significantly, discrepancies may result in the counts of participating ad serving system.

2.3.5 VAST Wrapper Ads

Wrapper ads provide a way for one ad server to redirect a video player to another, secondary ad server to retrieve an ad, multiple ads, or yet another VAST Wrapper.

One ad server may redirect to another for a variety of reasons:

- The first ad server has selected a specific advertiser campaign to fill the inventory. In this case the redirect instructs the secondary ad server to return specific ads from a particular ad campaign.
- The first ad server is delegating a specific piece of inventory for either a single ad or an entire Pod of ads to the secondary ad server to fill with any ads that are within an established agreement between the two parties.
- An ad server may have no Ad to return and may return a redirect to a backfill provider.

2.3.5.1 Infinite Loops and Dead Ends

When serving an ad involves a chain of wrappers, an infinite loop is possible where a chain of Wrappers never results in a final inline VAST response. Another case involves a finite number of wrappers in which the resulting inline response is used as a decisioning mechanism to find an ad instead of delivering the ad as required. In these cases, the decisioning mechanism may never return an ad or may take too long to return the ad.

In general, wrappers should be limited to five before resulting in an inline response. If the player detects more than five wrappers, the player may reject any subsequent responses in the chain, replace the [ERRORCODE] macro in the VAST/Ad/Wrapper/Error URI if provided to indicate that the wrapper limit was reached, and move on to the next option for an ad. Error codes should be sent for all wrappers in the chain where provided.

VPAID 4.0 offers timeout settings to prevent an ad from taking too long to load. When an inline response fails to produce an ad within the timeframe identified in VPAID or other ad framework, the player may reject the ad, send error code 304 to indicate that no ad was produced in the given timeframe, and move on to the next option for an ad. Error codes should also be sent to any wrappers preceding the inline response.

2.3.5.2 Wrapper Conflict Management and Precedence

When Companion creative are included directly in the wrapper response, conflict may occur. In a VAST ad, whether served with multiple wrappers or in one Inline response, all creative offered is intended to be part of the same creative concept, and the video player should attempt to display all creative presented in the response (or in a chain of responses). However, when conflict occurs, the video player should favor creative offered closest to the inline response.

For example, if a wrapper contains companion creative and the inline response also contains companion creative, the companion creative in the Inline response should be selected (unless both creative can be displayed without conflict).

In another example, if the inline response is absent of any companion creative but two or more wrappers contain companion creative, then creative for the wrapper served closest to the inline response should be favored. However, if multiple creative can be served without conflict, the video player should attempt to display whatever creative it can.

2.3.6 Error Reporting

The `<Error>` element enables the video player to provide feedback to ad servers when an Ad cannot be served. In VAST 3.0, detailed error codes and specifications for format are provided to enable detailed error logging for better ad serving diagnostics.

Providing more detailed error codes enables stronger diagnostics and enables better technology development over time. If ad servers can collect more detailed information about why their ads or specific creative couldn't be served, they can improve their systems to produce fewer errors.

The `<Error>` element is an optional element nested within the `<InLine>` or `<Wrapper>` element. It is used to track errors for an Ad. An error for an Inline Ad that is part of a chain of wrapper ads will produce an error for each of the wrappers used to serve the Inline Ad.

An `<Error>` element is also provided at the root VAST level and is primarily used to report a "No Ad" response. See section 2.3.6.4 for more information.

2.3.6.1 Ad Server Details: <Error> Element

An `<Error>` element includes a URI that provides a tracking resource for the error. This error-tracking resource is called when the video player is unable to display the Ad.

The following example is a sample VAST response that includes the <Error> element for an Inline Ad.

```
<InLine>
      …
      <Error>
            <![CDATA[http://adserver.com/error.gif]>
      </Error>
      …
</InLine>
```

If the ad server wants to collect more specific details about the error from the video player (as listed in section 2.3.6.3), an [ERRORCODE] macro can be included in the URI.

2.3.6.2 Video Player Details

If an error occurs while trying to load an Ad and the <Error> element is provided, the video player must:

- Request the error source file using the URI provided.

Replace the [ERRORCODE] macro, if provided, with the appropriate error code listed in the table in section 2.3.6.3. At a minimum, error code 900 (Unidentified error) can be used, but a more specific error code benefits all parties involved.

If the Ad was served after a chain of Wrapper Ad responses, the video player must also return error details as listed above for each Wrapper response that also includes error parameters. Macro responses must be correctly percent-encoded per RFC 3986.

The following table lists VAST error codes and their descriptions.

2.3.6.3 VAST Error Codes Table

Code	Description
100	XML parsing error.
101	VAST schema validation error.
102	VAST version of response not supported.
200	Trafficking error. Video player received an Ad type that it was not expecting and/or cannot display.
201	Video player expecting different linearity.
202	Video player expecting different duration.
203	Video player expecting different size.
204	Ad category was required but not provided.
300	General Wrapper error.

Code	Description
301	Timeout of VAST URI provided in Wrapper element, or of VAST URI provided in a subsequent Wrapper element. (URI was either unavailable or reached a timeout as defined by the video player.)
302	Wrapper limit reached, as defined by the video player. Too many Wrapper responses have been received with no InLine response.
303	No VAST response after one or more Wrappers.
304	InLine response returned ad unit that failed to result in ad display within defined time limit.
400	General Linear error. Video player is unable to display the Linear Ad.
401	File not found. Unable to find Linear/MediaFile from URI.
402	Timeout of MediaFile URI.
403	Couldn't find MediaFile that is supported by this video player, based on the attributes of the MediaFile element.
405	Problem displaying MediaFile. Video player found a MediaFile with supported type but couldn't display it. MediaFile may include: unsupported codecs, different MIME type than `MediaFile@type`, unsupported delivery method, etc.
406	Mezzanine was required but not provided. Ad not served.
407	Mezzanine is in the process of being downloaded for the first time. Download may take several hours. Ad will not be served until mezzanine is downloaded and transcoded.
408	Conditional ad rejected.
409	Interactive unit in the InteractiveCreativeFile node was not executed.
410	Verification unit in the Verification node was not executed.
411	Mezzanine was provided as required, but file did not meet required specification. Ad not served.
500	General NonLinearAds error.
501	Unable to display NonLinear Ad because creative dimensions do not align with creative display area (i.e. creative dimension too large).
502	Unable to fetch NonLinearAds/NonLinear resource.
503	Couldn't find NonLinear resource with supported type.

Code	Description
600	General CompanionAds error.
601	Unable to display Companion because creative dimensions do not fit within Companion display area (i.e., no available space).
602	Unable to display required Companion.
603	Unable to fetch CompanionAds/Companion resource.
604	Couldn't find Companion resource with supported type.
900	Undefined Error.
901	General VPAID error.

2.3.6.4 No Ad Response

When the ad server does not or cannot return an Ad, the VAST response should contain only the root `<VAST>` element with optional `<Error>` element, as shown below:

```
<VAST version="4.0">
      <Error>
            <![CDATA[http://adserver.com/noad.gif]>
      </Error>
</VAST>
```

The VAST `<Error>` element is optional but if included, the video player must send a request to the URI provided when the VAST response returns an empty InLine response after a chain of one or more wrapper ads. If an `[ERRORCODE]` macro is included, the video player should substitute with error code `303`.

Besides the VAST level `<Error>` resource file, no other tracking resource requests are required of the video player in a no-ad response in either the Inline Ad or any Wrapper ads.

2.3.7 Macros

Sometimes ad servers would like to collect metadata from the video player when tracking event URIs are accessed. For example, the position of the video player playhead at the time a tracking event URI is accessed is useful to the ad server and is data that can only be known at the time of the prescribed tracking event. This data cannot be built into the URI at the time the VAST response is built and served.

The following macros enable the video player to provide certain details to the ad server at the time tracking URIs are accessed.

[ERRORCODE]: replaced with one of the error codes listed in section 2.3.6.3 when the associated error occurs; reserved for error tracking URIs.

[CONTENTPLAYHEAD]: replaced with the current time offset "`HH:MM:SS.mmm`" of the video content.

[CACHEBUSTING]: replaced with a random 8-digit number.

[ASSETURI]: replaced with the URI of the ad asset being played.

[TIMESTAMP]: the date and time at which the URI using this macro is accessed. Used where ever a time stamp is needed, the macro is replaced with the date and time using the formatting conventions of ISO 8601. However, ISO 8601 does not provide a convention for adding milliseconds. To add milliseconds, use the convention .mmm at the end of the time provided and before any time zone indicator. For example, January 17, 2016 at 8:15:07 and 127 milleseconds, Eastern Time would be formatted as follows: 2016-01-17T8:15:07.127-05

When replacing macros, the video player must correctly percent-encode any characters as defined by RFC 3986.

VAST doesn't provide any guidance on URI format, but using the `[CACHEBUSTING]` macro simplifies trafficking, enabling ad servers to easily search and replace the appropriate macro for cache busting.

2.3.8 Industry Icon Support

Several initiatives in the advertising industry involve using an icon that overlays on top of an Ad creative to provide some extended functionality such as to communicate with consumers or otherwise fulfill requirements of a specific initiative. Often this icon and its functionality may be provided by a vendor, and is not necessarily served by the ad server or included in the creative itself.

One example of icon use is for compliance to certain Digital Advertising Alliance (DAA) self-regulatory principles for interest-based advertising (IBA). This section provides an overview of how video players can support the use of icons in a general manner while using the DAA's AdChoices program, as a specific example.

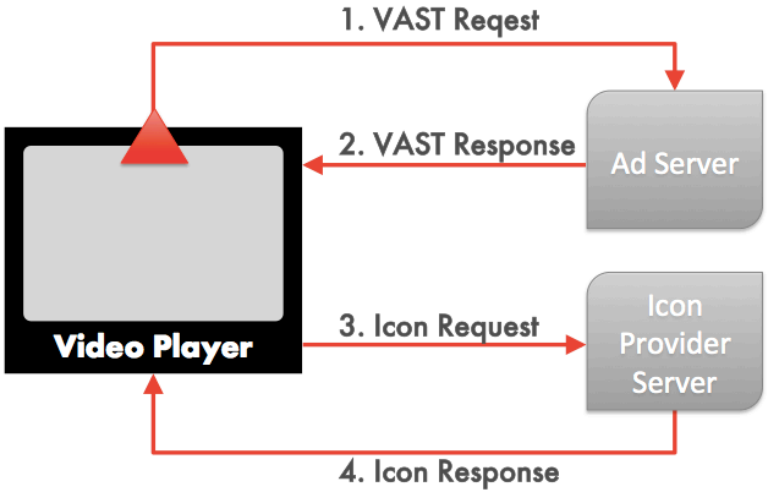
2.3.8.1 Icon Use Case: AdChoices for Interest-Based Advertising (IBA)

The DAA sets forth principles that endeavor to give consumers a better understanding of and greater control over ads that are customized based on the consumer's online behavior. This control is made available to the consumer in the form of the AdChoices icon, which is displayed in a prominent location in or around the Ad creative. When a consumer clicks the icon, they may be offered: information about the ad server and data providers used to select the Ad, options to learn more about OBA, and the ability for consumers to opt out from receiving OBA ads in the future.

2.3.8.2 The <Icons> Element

VAST 3.0 introduced the `<Icons>` element, which is offered under the `<Linear>` creative element for both Inline and Wrapper ads.

The following diagram illustrates the general process for how the <Icons> element is represented in a VAST response.



The Icon Provider Server represented in this diagram may be the same server that serves the VAST response but more commonly, is a vendor that serves the icon from its own systems.

When the <Icons> element is included in the VAST response, the video player must display the object as an overlay on top of the Linear Ad with which the icon is served and after the ad video has started (i.e. first frame of video is displayed in the player).

Video Player Implementation Note	Since a vendor often serves icons and may charge advertising parties for each icon served, the video player should not pre-fetch the icon resource until the resource can be displayed. Pre-fetching the icon resource may cause the icon provider to falsely record an icon view when the icon may not have been displayed.

2.3.8.3 Precedence and Conflict Management:

As an Ad goes through a delivery chain, companies may include their own Icon element in their wrapper responses. Sometimes these multiple icon elements are all for the same program and the video player must decide on only one icon to display. When icon elements represent more than one program, one icon from each program should be displayed.

The video player can use its own business rules to decide which icon to display, along with any specific program recommendations. For example, when multiple AdChoices icons are offered, the DAA program recommendation is to select the icon that is closest to the creative. To comply with the AdChoices program when multiple AdChoices icons are served, the video player must choose the icon closest to the creative.

If no other rules govern the selection of which icon to display, the video player should choose the one closest to the creative. That is, if the <Icon> element is included within the Inline Ad, then that icon is the closest to the creative. However, if the Inline Ad contains no

<Icon> element, but the last Wrapper Ad in a chain of Wrappers did contain the <Icon> element, then the icon from that last Wrapper Ad is the one closest to the creative.

When multiple icons from more than one icon program are included in a chain of Wrapper ads, the video player must decide which icon from each program should be displayed. Again, the video player can use its own business rules; however, the icons must not overlap each other. If all program icons use the same xPosition and yPosition values, the video player can use width and height attribute values to offset coordinates relative to the display area of the Ad creative.

Video Player Implementation Note	A video player may not be able to display an Icon but should make every attempt to do so.

2.4 Viewability Verification and Interactive Linear Creative

VAST 4.0 adds new sections in the Linear file for viewability, ad verification, and interactive creative files. These new sections offer performance and measurement benefits but also add a level of complexity.

Player compliance with VAST 4.0 requires appropriate execution of these files. In general, any verification code should be executed before the ad is loaded and any interactive creative files should be executed next and before the ad is loaded.

Player expectations on these added features are summarized here and further defined in their corresponding sections.

2.4.1 Publisher Viewability

Publishers have the option to offer viewable impression tracking on the ad using the <ViewableImpression> feature added in VAST 4.0. Three URIs may be provided to track whether the ad was <Viewable>, <NotViewable>, or <ViewUndetermined>. Requirements for viewability are details the buyer and seller should negotiate per transaction; however, at a minimum, or in the absence of such agreement, the publisher should follow the MRC guidelines for viewable video ad impressions. At the release of this document, the general requirements for viewable video ad impressions could be found on page 8 of the MRC Viewable Ad Impression Guidelines, Version 2.0 (Final with 2015 additions).

2.4.2 Viewability with Ad Verification Services

Ad Verification services can use the <AdVerifications> feature in VAST 4.0 to place their ad verification code. More than one vendor may use this feature. Each vendor uses a different <Verification> node to place code that is either a <JavaScriptResource> or a <FlashResource>.

A VAST 4.0 player must check for code in these sections of the VAST file and attempt to execute any verification files *before* the linear ad creative is executed. If the verification code cannot be executed, an <Error> URI may be sent.

The <ViewableImpression> feature under <Verification> is for the ad verification vendor to use as needed. The player need not use this feature unless prior arrangements have been made to do so.

2.4.3 Interactive Linear Creative Files

In VAST 4.0, the `<MediaFile>` should only be used to include video files. For linear files that require an API framework to be executed, the new `<InteractiveCreativeFile>` should be used to include these files. Once the `<AdVerifications>` element has been checked for verification code, the `<InteractiveCreativeFile>` element should be checked for code in order to execute the ad. When the `<InteractiveCreativeFile>` cannot be executed, the `<Error>` should be sent and the player may check the `<MediaFiles>` element for any videos that are available to be played.

3 VAST Implementation

VAST is an XML schema for providing metadata about an ad for in-stream video that is parsed by a player or by a server on the player's behalf. This section provides the details for forming the VAST.

Beginning with section 3.1, each element available in VAST is described and a table summarizes information about hierarchy, requirements, and attributes. Each VAST element that includes nested elements is defined under a second-level heading in this document. Third-level headings represent nested elements that have no additional nested elements under them.

Links to the table of contents (TOC) and the schema are provided under each heading to aid in navigation. The human-readable schema in section summarizes VAST elements and provides a link to the chapter that describes the element in more detail.

Before forming a VAST document, considerations for the XML namespace and browser security for Flash and JavaScript should be established as follows.

XML Namespace

Whenever VAST is used in conjunction with any other XML template, such as with VMAP or VAST extensions, a namespace should be declared for each so that the elements of one are not confused with the elements of another.

For more information, visit: http://www.w3.org/TR/REC-xml-names/

Browser Security for Flash™ and JavaScript™

Modern browsers restrict Adobe Flash and JavaScript runtime environments from retrieving data from other servers. Since typical VAST responses come from other servers, measures must be taken for each:

crossdomain.xml for Flash

To enable Flash video players to accept a VAST response, ad servers must provide a crossdomain.xml file at their root HTTP domain. For example, adserver.com should provide the file as follows:

```
http://adserver.com/crossdomain.xml
```

Flash video players know to check the root domain of any server that sends it data.

The xml is a simple file that includes the following:

```
<cross-domain-policy>
      <allow-access-from domain="*”>
<cross-domain-policy>
```

The asterisks (*) value for the domain attribute enables any Flash video player to accept data from the server providing the crossdomain.xml file.

For more information, visit http://kb2.adobe.com/cps/142/tn_14213.html

Cross Origin Resource Sharing (CORS) for JavaScript
In order for JavaScript video players to accept a VAST response, ad servers must include a CORS header in the http file that wraps the VAST response. The CORS header must be formatted as follows:

```
Access-Control-Allow-Origin: <origin header value>
Access-Control-Allow-Credentials: true
```

These HTTP headers allow an ads player on any origin to read the VAST response from the ad server origin. The value of `Access-Control-Allow-Origin` should be the value of the `Origin` header sent with the ad request.

Setting the `Access-Control-Allow-Credentials` header to `true` will ensure that cookies will be sent and received properly.

For more information, visit http://www.w3.org/TR/cors

3.1 Declaring the VAST response

All VAST responses share the same general structure. Each VAST response is declared with `<VAST>` as its topmost element along with the `version` attribute indicating the official version with which the response is compliant. For example, a VAST 4.0 response is declared as follows:

```
<VAST version="4.0">
```

As with all XML documents, each element must be closed after details nested within the element are provided. The following example is a VAST response with one nested `<Ad>` element.

```
<VAST version="4.0">
      <Ad>
            <!--ad details go here-->
      </Ad>
</VAST>
```

3.2 VAST

TOC Schema

VAST is the root node for a VAST-compliant ad response and is used to declare the VAST response as described in section 3.1.

Required	Yes
Parent	None (root)
Bounded	1
Sub-elements	Error Ad
Attributes	Description
version	A float number (number with decimal) to indicate the VAST version being used.

3.2.1 Error (VAST)

TOC Schema

Used when in a no ad response. When the ad server does not or cannot return an Ad, the VAST response should contain only the root `<VAST>` element with optional `<Error>` element, as shown below:

```
<VAST version="4.0">
      <Error>
            <![CDATA[http://adserver.com/noad.gif]>
      </Error>
</VAST>
```

The VAST `<Error>` element is optional but if included, the video player must use the URI provided to notify the server that no ad was returned. Any other tracking URI provided outside the VAST `<Error>` element may be ignored.

Required	No
Parent	VAST
Bounded	0-1
Sub-elements	none

3.3 Ad

TOC Schema

The `<Ad>` element may contain an `<InLine>` ad or a `<Wrapper>`. The wrapper points to a secondary server for another VAST response, which may be another wrapper or an inline response. An inline response contains the ad creative necessary to execute ad playback.

3.3.1 Ad Pods and Stand-Alone Ads

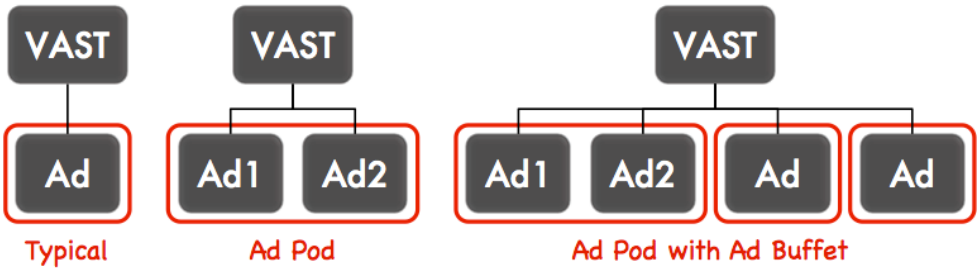
TOC Schema

While a single `<Ad>` element represents the most common VAST response, multiple ads may be included as either stand-alone ads or a pod of ads, or a mix of both. Ads in a pod are distinguished by using the `sequence` attribute for an `<Ad>`, denoting which ad plays first, second, and so on. If the player supports ad pods, sequenced ads are played in numerical

order and all ads in the pod should be played to the best of the player's ability. All `sequence` values in a VAST response must be unique.

Non-sequenced ads, are stand-alone ads and considered part of an "ad buffet" from which the player may select one or more ad to play in any order. Stand-alone ads may be included in a VAST response with an ad pod and may be used to substitute an ad in the pod when an ad cannot be played.

The following diagram illustrates some options for how the `<Ad>` element may be represented in a VAST response.



If the video player cannot display an entire ad pod or any stand-alone ads, it can decline from loading the ad resources and use the error URI, if provided, to notify the server.

Playing a Pod of Ads

When electing to play a pod of ads returned by the ad server, the video player must play the ads in the Pod in the prescribed sequence and should play as many of the ads as possible. The player may elect to truncate any ads at the end of an ad pod if either: the ads cannot be played because they cannot physically fit into the ad break in the stream (such as when time is limited in a live stream) or if the pod violated any limits specified by the video player request (for example: number of ads to return, or maximum pod duration).

When an ad pod is the result of following a VAST `<Wrapper>` the same impression and tracking URIs in the VAST `<Wrapper>` are called as each ad in the pod is played.

Should an ad in the pod fail to play, the video player should substitute an un-played stand-alone ad from the response. If stand-alone ads are unavailable, the player should move on to the next ad in the ad pod.

Because each ad in an ad pod uses a URI to provide the ad creative, one or more of the ads in the pod may include a URI to a VAST wrapper. Wrappers should not be used in an ad pod. Not only could a wrapper trigger a chain of wrappers for the ad and slow performance, but also a wrapper could call a secondary ad pod to fill the slot for one ad in the original ad pod. This ad-pod-within-an-ad-pod situation can cause a "tree" of ads that becomes difficult or impossible to execute.

If a wrapper is used to provide an ad in the ad pod, the wrapper should use attributes, `allowMultipleAds=false` and `followAdditionalWrappers=false` to prevent performance issues that result from an unmanaged string of wrappers and multiple ads in an ad pod.

Video Player Implementation Note	If multiple `<Ad>` elements are provided with `sequence` attributes and the player supports ad pods, all ads in the pod must be played to the best of the player's ability. If not supported or the pod cannot be played, the video player should use the error-tracking URI, if provided, to notify the server. A special exemption exists when using VMAP. Please visit http://iab.com/vmap for information on VMAP.

3.3.2 The Ad Element

TOC Schema

Properties for the `<Ad>` element are listed in the following table.

Required	Yes (unless there is no ad to return; in which case `<Error>` should be used to provide an error URI)
Parent	VAST
Bounded	1+
Sub-elements	InLine* Wrapper*
Attributes	Description
id	an ad server-defined identifier string for the ad
sequence	a number greater than zero (0) that identifies the sequence in which an ad should play; all `<Ad>` elements with sequence values are part of a pod and are intended to be played in sequence
conditionalAd	A Boolean value that identifies a conditional ad. In the case of programmatic video ad serving, a VPAID ad unit or other mechanism might be used to decide whether there is an ad that matches the placement. When there is no match, an ad may not be served. Use of the `conditionalAd` attribute enables publishers to avoid accepting these ads in placements where an ad must be served. A value of `true` indicates that the ad is conditional and should be used in all cases where the inline executable unit (such as VPAID) is not an ad but is instead a framework for finding an ad; a value of `false` is the default value and indicates that an ad is available.

*one `<InLine>` or `<Wrapper>` element is required but both are not allowed

3.4 InLine

TOC Schema

Within the nested elements of an `<InLine>` ad are all the files and URIs necessary to play and track the ad. In a chain of `<Wrapper>` VAST responses, an `<InLine>` response ends the chain.

Required	One of `<InLine>` or `<Wrapper>` is required, but both are not allowed
Parent	Ad
Bounded	1
Sub-elements	AdSystem* AdTitle* Impression* Category Description Advertiser Pricing Survey Error Extensions ViewableImpression AdVerification Creatives*

*required

3.4.1 AdSystem

TOC Schema

The ad serving party must provide a descriptive name for the system that serves the ad. Optionally, a version number for the ad system may also be provided using the `version` attribute.

Required	Yes
Parent	InLine or Wrapper
Bounded	1
Content	A string that provides the name of the ad server that returned the ad
Attributes	Description
version	A string that provides the version number of the ad system that returned the ad

3.4.2 AdTitle

TOC Schema

The ad serving party must provide a title for the ad using the `<AdTitle>` element. If a longer description is needed, the `<Description>` element can be used.

Required	Yes
Parent	InLine
Bounded	1
Content	A string that provides a common name for the ad

3.4.3 Impression

TOC Schema

The ad server provides an impression-tracking URI for either the InLine ad or the Wrapper using the `<Impression>` element. All `<Impression>` URIs in the InLine response and any Wrapper responses preceding it should be triggered at the same time when the impression for the ad occurs, or as close in time as possible to when the impression occurs, to prevent impression-counting discrepancies.

Required	Yes
Parent	InLine or Wrapper
Bounded	1
Content	A URI that directs the video player to a tracking resource file that the video player must use to notify the ad server when the impression occurs.
Attributes	Description
id	An ad server id for the impression. Impression URIs of the same id for an ad should be requested at the same time or as close in time as possible to help prevent discrepancies.

3.4.4 Category

TOC Schema

Used in creative separation and for compliance in certain programs, a category field is needed to categorize the ad's content. Several category lists exist, some for describing site content and some for describing ad content. Some lists are used interchangeably for both site content and ad content. For example, the category list used to comply with the IAB Quality Assurance Guidelines (QAG) describes site content, but is sometimes used to describe ad content.

The VAST category field should only use AD CONTENT description categories.

The `authority` attribute is used to identify the organizational authority that developed the list being used. In some cases, the publisher may require that an ad category be identified. If required by the publisher and not provided, the publisher may skip the ad, notify the ad server using the `<Error>` URI, if provide (error code 203), and move on to the next option.

If category is used, the `authority=` attribute must be provided.

Required	No*
Parent	InLine
Bounded	0+
Content	A string that provides a category code or label that identifies the ad content.
Attributes	Description
authority *	A URL for the organizational authority that produced the list being used to identify ad content.

*Optional unless the publisher requires ad categories. The `authority` attribute is required if categories are provided.

3.4.5 Description

TOC Schema

When a longer description of the ad is needed, the `<Description>` element can be used.

Required	No
Parent	InLine
Bounded	0-1
Content	A string value that describes a longer description of the ad

3.4.6 Advertiser

TOC Schema

Providing an advertiser name can help publishers prevent display of the ad with its competitors. Ad serving parties and publishers should identify how to interpret values provided within this element.

Required	No
Parent	InLine
Bounded	0-1
Content	A string that provides the name of the advertiser as defined by the ad serving party.

3.4.7 Pricing

TOC Schema

Used to provide a value that represents a price that can be used by real-time bidding (RTB) systems. VAST is not designed to handle RTB since other methods exist, but this element is offered for custom solutions if needed. If the value provided is to be obfuscated or encoded, publishers and advertisers must negotiate the appropriate mechanism to do so.

When included as part of a VAST Wrapper in a chain of Wrappers, only the value offered in the first Wrapper need be considered.

Required	No
Parent	InLine or Wrapper
Bounded	0-1
Content	A numerical value that represents a price that can be used in real-time bidding systems
Attributes	Description
model*	Identifies the pricing model as one of: CPM, CPC, CPE, or CPV
currency*	The three-letter ISO-4217 currency symbol that identifies the currency of the value provided (e.g. USD, GBP, etc.)

*required

3.4.8 Survey

TOC Schema

Ad tech vendors may want to use the ad to collect data for resource purposes. The `<Survey>` element can be used to provide a URI to any resource file having to do with collecting survey data. Publishers and any parties using the `<Survey>` element should determine how surveys are implemented and executed. Multiple survey elements may be provided.

A `type` attribute is available to specify the MIME type being served. For example, the attribute might be set to `type="text/javascript"`. Surveys can be dynamically inserted into the VAST response as long as cross-domain issues are avoided.

Required	No
Parent	InLine
Bounded	0+
Content	A URI to any resource relating to an integrated survey.
Attributes	Description
type	The MIME type of the resource being served.

3.4.9 Error (InLine and Wrapper)

TOC Schema

The `<Error>` element contains a URI that the player uses to notify the ad server when errors occur with ad playback. If the URI contains an `[ERRORCODE]` macro, the video player must populate the macro with an error code as defined in section 2.3.7. If no specific error can be found, error 900 may be used to indicate an undefined error; however, every attempt should be made to provide an error code that maps to the error that occurred. The `<Error>` element is available for both the InLine or Wrapper elements.

Required	No
Parent	InLine or Wrapper
Bounded	0+
Content	A URI to a tracking resource to be used when an error in ad playback occurs.

3.5 ViewableImpression

TOC Schema

The ad server may provide impression-tracking URIs for both the InLine ad and any Wrappers using the `<ViewableImpression>` element. Tracking URIs may be provided in three containers: `<Viewable>`, `<NotViewable>`, and `<ViewUndetermined>`.

The point at which these tracking resource files are pinged depends on the viewability standards against which the publisher is certified or any alternate standards document for the transaction between the publisher and advertiser. At the time of this document's release, the Media Ratings Council (MRC) had published video viewability recommendations for counting a video ad view after 50% of the ad's pixels are in view for at least two seconds. Publishers should disclose their process for tracking viewable video impressions.

General Implementation Note	The <ViewableImpression> element available for InLine and Wrapper ads is available for the publisher to support any viewable-tracked inventory. A secondary <ViewableImpression> element is available for verification vendors under the <Verification> element ad described in section 3.17.3.

Player support for the <ViewableImpression> element is optional. When used, URIs for the inline ad as well as any wrappers used to serve the ad should all be triggered at the same time, or as close in time as possible to when the criteria for a viewable video impression is met.

Required	No
Parent	Inline or Wrapper
Bounded	0-1
Sub-elements	Viewable NotViewable ViewUndetermined
Attributes	Description
id	An ad server id for the impression. Viewable impression resources of the same id should be requested at the same time, or as close in time as possible, to help prevent discrepancies.

3.5.1 Viewable

TOC Schema

The <Viewable> element is used to place a URI that the player triggers if and when the ad meets criteria for a viewable video ad impression.

Required	No
Parent	ViewableImpression
Bounded	0+
Content	A URI that directs the video player to a tracking resource file that the video player should request at the time that criteria is met for a viewable impression.

3.5.2 NotViewable

TOC Schema

The <NotViewable> element is a container for placing a URI that the player triggers if the ad is executed but never meets criteria for a viewable video ad impression.

Required	No
Parent	ViewableImpression
Bounded	0+
Content	A URI that directs the video player to a tracking resource file that the video player should request if the ad is executed but never meets criteria for a viewable impression.

3.5.3 ViewUndetermined

TOC Schema

The `<ViewUndetermined>` element is a container for placing a URI that the player triggers if it cannot determine whether the ad has met criteria for a viewable video ad impression.

Required	No
Parent	ViewableImpression
Bounded	0+
Content	A URI that directs the video player to a tracking resource file that the video player should request if the player cannot determine whether criteria is met for a viewable impression.

3.6 Creatives

TOC Schema

The `<Creatives>` (plural) element is a container for one or more `<Creative>` (singular) element used to provide creative files for the ad. For an InLine ad, the `<Creatives>` element nests all the files necessary for executing and tracking the ad.

In a Wrapper, elements nested under `<Creatives>` are used mostly for tracking. Companion and Icon creative may be included in a wrapper, but files for Linear and NonLinear ads can only be provided in the InLine version of the ad.

Required	Yes
Parent	InLline or Wrapper
Bounded	1
Sub-elements	Creative

*required

3.7 Creative

TOC Schema

Each `<Creative>` element contains nested elements that describe the type of ad being served using nested sub-elements for Linear, NonLinearAds, and/or CompanionAds. Multiple creative may be used to define different components of the ad. At least one `<Linear>` or `<NonLinearAds>` element is required under the Creative element.

Required	Yes
Parent	Creatives for both InLine and Wrapper formats
Bounded	1+
Sub-elements	UniversalAdId* CreativeExtensions Linear NonLinearAds CcompanionAds
Attributes	Description
id	A string used to identify the ad server that provides the creative.
adId	Used to provide the ad server's unique identifier for the creative. In VAST 4.0, the UniversalAdId element was introduced to provide a unique identifier for the creative that is maintained across systems. Please see section 3.7.1 for details on the UniversalAdId.
sequence	A number representing the numerical order in which each sequenced creative within in ad should play.
apiFramework	A string that identifies an API that is needed to execute the creative.

*required

General Implementation Note	The Creative `sequence` attribute should not be confused with the Ad `sequence` attribute. Creative `sequence` identifies the sequence of multiple creative within a single ad and does NOT define a pod. Conversely, the Ad `sequence` identifies the sequence of multiple ads and defines an ad pod. See section 3.3.1 for details about ad pods.

3.7.1 UniversalAdId

TOC Schema

A required element for the purpose of tracking ad creative, the `<UniversalAdId>` is used to provide a unique creative identifier that is maintained across systems. This creative ID may be generated with an authoritative program, such as the AD-ID® program in the United States, or by any company's creative identification system. Some countries may have specific requirements for ad-tracking programs.

The UniversalAdId element is required in 4.0, but the attribute values for `idRegistry` and `idValue` are to be used to support a company's need for tracking ad creative. If no common registry is used, a company may use its own creative identification system with a custom `idRegistry` to support that system as long as the `idValue` can be maintained across systems. Ad servers and publishers should discuss what is required for this field to support a successful ad campaign.

NOTE: a creative id can also be included in the `adId` attribute used in the `<Creative>` element, but that creative id should be used to specify the ad server's unique identifier. The UniversalAdId is used for maintaining a creative id for the ad across multiple systems.

Required	Yes
Parent	Creative only in the InLine format
Bounded	1
Content	A string identifying the unique creative identifier.
Attributes	Description
idRegistry*	A string used to identify the URL for the registry website where the unique creative ID is cataloged. Default value is "unknown."
idValue*	A string for the unique creative ID. Default value is "unknown."

*required

3.7.2 CreativeExtensions

TOC Schema

When an executable file is needed as part of the creative delivery or execution, a `<CreativeExtensions>` element can be added under the `<Creative>`. This extension can be used to load an executable creative with or without using the `<MediaFile>`.

A `<CreativeExtension>` (singular) element is nested under the `<CreativeExtensions>` (plural) element so that any XML extensions are separated from VAST xml. Additionally, any XML used in this extension should identify an XML name space (xmlns) to avoid confusing any of the extension element names with those of VAST.

The nested `<CreativeExtension>` includes an attribute for `type`, which specifies the MIME type needed to execute the extension.

Required	No
Parent	Creative only in the InLine format
Bounded	0+
Sub-elements	CreativeExtension

3.7.3 CreativeExtension

TOC Schema

Used as a container under the CreativeExtentions element, this node is used to delineate any custom XML object that might be needed for ad execution.

Required	No
Parent	CreativeExtensions only in the InLine format
Bounded	0+
Content	Custom XML object
Attributes	Description
type	The MIME type of any code that might be included in the extension.

3.8 Linear

TOC Schema

Linear ads are the video-formatted ads that play linearly within the streaming content, meaning before the streaming content, during a break, or after the streaming content. A Linear creative contains a `<Duration>` element to communicate the intended runtime and a `<MediaFiles>` element used to provide the needed video files for ad execution.

Required	At least one of Linear or NonLinearAds is required.
Parent	Creative for both InLine and Wrapper formats
Bounded	0-1
Sub-elements	Duration* MediaFiles* InteractiveCreativeFile AdParameters TrackingEvents VideoClicks Icons
Attributes	Description
skipOffset	A time value that identifies when skip controls should be made available to the end user; publisher may define a minimum `skipOffset` value in its policies and disregard Skippable creative when `skipoffset` values are lower than publisher's minimum.

*required

3.8.1 Duration

TOC Schema

The ad server uses the `<Duration>` element to denote the intended playback duration for the Linear component of the ad. Time value may be in the format HH:MM:SS.mmm where .mmm indicates milliseconds. Providing milliseconds is optional.

Required	Yes (when Linear is used)
Parent	Linear only in the InLine format
Bounded	0-1
Content	A time value for the duration of the Linear ad in the format HH:MM:SS.mmm (.mmm is optional and indicates milliseconds).

3.8.2 AdParameters

TOC Schema

Some ad serving systems may want to send data to the media file when first initialized. For example, the media file may use ad server data to identify the context used to display the creative, what server to talk to, or even which creative to display. The optional `<AdParameters>` element for the Linear creative enables this data exchange.

The optional attribute `xmlEncoded` is available for the `<AdParameters>` element to identify whether the ad parameters are xml-encoded. If true, the video player can only decode the data using xml. Video players operating on earlier versions of VAST may not be able to xml-

decode data, so data should only be xml-encoded when being served to video players capable of xml-decoding the data.

When a VAST response is used to serve a VPAID ad unit, the `<AdParameters>` element is currently the only way to pass information from the VAST response into the VPAID object; no other mechanism is provided.

Required	No
Parent	Linear only in the InLine format NonLinear only in InLine format Companion for both InLine and Wrapper formats
Bounded	0-1
Content	Metadata for the ad.
Attributes	Description
xmlEncoded	Identifies whether the ad parameters are xml-encoded.

*required

3.9 MediaFiles

TOC Schema

Since the first version of VAST, the MediaFiles element was designated for linear video files. Over the years as digital video technology advances, the media files placed in a VAST tag have come to include complex files that require interactive API integration. Players not equipped with the technology to execute such files may be unable to play the ad or execute interactive components. In version 4.0, the linear video files should be separate from the interactive components that require API integration. Separating these files enable the ad to play in more players and improves ad play performance.

Linear media files should be submitted as follows:

Video file only: Include three `<MediaFile>` elements (section 3.9.1), each with a URI to a ready-to-serve video file at quality levels for high, medium, and low. Please review the IAB Digital Video Ad Format Guidelines for guidance on ready-to-serve file quality specifications.

Video file for use in ad-stitching: In addition to the three ready-to-serve files, use the `<Mezzanine>` element (section 3.9.2) to include a URI to the raw video file. Please review the IAB Digital Video Ad Format Guidelines for guidance on mezzanine file specifications.

Interactive linear video file: In addition to at least one ready-to-serve video file included in the `<MediaFile>` element, use the `<InteractiveCreativeFile>` element (section 3.9.3) to include a URI to the interactive media file, specifying the API framework required to execute the file. When interactive files are included in the VAST response, they should be executed *before* any video files are executed.

The components of the <MediaFiles> elements:

Required	Yes (Linear ads)
Parent	Linear only for InLine format
Bounded	1
Sub-elements	MediaFile* Mezzanine** InteractiveCreativeFile

*required **required in ad-stitched video executions

3.9.1 MediaFile

TOC Schema

In VAST 4.0 `<MediaFile>` should only be used to contain the video file for a linear ad. In particular, three ready-to-serve files should be included, each of a quality level for high, medium, or low. A ready-to-serve video file is a video that is transcoded to a level of quality that can be transferred over an internet connection within a reasonable time for viewing. Each ready-to-serve file must be of the same MIME type and, if different MIME types files are made available for the ad, three ready-to-serve files should represent each MIME type separately.

When VPAID or another interactive API is needed to deliver and execute the linear ad, the URI to the VPAID or interactive file should be included in the `<InteractiveCreativeFile>`. In addition, at least one non-VPAID ready-to-serve video ad should be available in `<MediaFile>`. A VPAID or interactive video ad may also be included in a separate `<MediaFile>` element for execution in players that only support previous versions of VAST.

Guidelines for ad files that fulfill quality levels of high, medium, or low can be found in the IAB Digital Video Ad Format Guidelines. An adaptive bitrate streaming file featuring files at the three quality levels may also be provided.

Required	Yes
Parent	MediaFiles only for InLine format
Bounded	1+
Content	A CDATA-wrapped URI to a media file.
Attributes	Description
delivery*	Either “progressive” for progressive download protocols (such as HTTP) or “streaming” for streaming protocols.
type*	MIME type for the file container. Popular MIME types include, but are not limited to “video/x-flv” for Flash Video and “video/mp4” for MP4.
width*	The native width of the video file, in pixels.
height*	The native height of the video file, in pixels.
codec	The codec used to encode the file which can take values as specified by RFC 4281: http://tools.ietf.org/html/rfc4281.
id	An identifier for the media file.
bitrate or minBitrate and maxBitrate	For progressive load video, the `bitrate` value specifies the average bitrate for the media file; otherwise the `minBitrate` and `maxBitrate` can be used together to specify the minimum and maximum bitrates for streaming videos.

scalable	a Boolean value that indicates whether the media file is meant to scale to larger dimensions.
maintainAspectRatio	a Boolean value that indicates whether aspect ratio for media file dimensions should be maintained when scaled to new dimensions.
apiFramework**	identifies the API needed to execute an interactive media file, but current support is for backward compatibility. Please use the `<InteractiveCreativeFile>` element to include files that require an API for execution.

*required
**if an API framework is needed to execute the ad, please use `<InteractiveCreativeFile>` to provide API files.

3.9.2 Mezzanine

TOC Schema

The video player may use the raw mezzanine file to transcode video files at quality levels specific to the needs of certain environments. An XSD will validate this element as optional, but a mezzanine file is required in ad-stitched executions and whenever a publisher requires it. If no mezzanine file is available, this element may be left blank; however, publishers that require it may ignore the VAST response when not provided. If an ad is rejected for this reason, error code 406 is available to communicate the error when an `<Error>` URI and macro are provided.

The mezzanine file is used to transcode a file that can play in the systems they support and should never be used for direct ad playback.

The mezzanine file specifications are defined in the Digital Video Ad Format Guidelines.

Required	No*
Parent	MediaFile only in InLine format
Bounded	0-1
Content	A CDATA-wrapped URI to a raw, high-quality media file.

* VAST tags served to ad-stitching servers require a mezzanine file and may reject the VAST response if no mezzanine file is provided.

3.9.3 InteractiveCreativeFile

TOC Schema

For any media file that uses interactive APIs for advanced creative functionality, the `<InteractivityCreativeFile>` element is used to identify the file and the framework needed for execution. For example, when VAST uses VPAID to deliver and execute the ad, the VPAID unit should be contained in the `<InteractiveCreativeFile>` element.

Providing the interactive portion for a media file in a section of VAST separate from the video file enables players to more easily play the video file when no support is available to execute the API, especially for players that work with an ad-stitching service or make ad calls from a server on behalf of the player.

The player should attempt to execute the interactive file before attempting to load any `<MediaFile>` videos, but if the file cannot be executed, the player should trigger any included error URIs and use error code 409 when macros are provided.

Required	No
Parent	MediaFile
Bounded	1
Content	A CDATA-wrapped URI to a file that provides creative functions for the media file.
Attributes	Description
type	Identifies the MIME type of the file provided.
apiFramework	identifies the API needed to execute the Icon resource file if applicable

3.10 VideoClicks

TOC Schema

The `<VideoClicks>` element provides URIs for `clickthroughs`, `clicktracking`, and `custom` clicks and is available for Linear ads in both the InLine and Wrapper formats. Both InLine and Wrapper formats offer the ClickTracking and CustomClick elements, but only the InLine Linear ad offers the Clickthrough element. These elements are defined in the following sections.

Required	No
Parent	Linear in both the InLine and Wrapper format
Bounded	0-1
Sub-elements	Clickthrough (only available for InLine Linear ads) CickTracking CustomClick

3.10.1 ClickThrough

TOC Schema

The <`ClickThrough`> is a URI to the advertiser's site that the video player opens when a viewer clicks the ad. The clickthrough is only available in the InLine format and is used when the linear ad unit cannot handle a clickthrough.

Required	One ClickThrough element is required if `<VideoClicks>` in the InLine format is used.
Parent	VideoClicks only in the InLine format
Bounded	1+ (if `<VideoClicks>` is used)
Content	a URI to the advertiser's site that the video player opens when a viewer clicks the ad.

3.10.2 ClickTracking

TOC Schema

Multiple `<ClickTracking>` elements can be used in the case where multiple parties would like to track the linear ad clickthrough.

Required	No
Parent	Linear in both InLine and Wrapper formats.
Bounded	0+
Content	A URI for tracking when the ClickThrough is triggered.
Attributes	Description
id	A unique ID for the click to be tracked

3.10.3 CustomClick

TOC Schema

The `<CustomClick>` is used to track any interactions with the linear ad that do not include the clickthrough click and do not take the viewer away from the video player. For example, if an ad vendor wants to track that a viewer clicked a button to change the ad's background color, the `<CustomClick>` element holds the URI to notify the ad vendor that this click happened. An API may be needed to inform the player that a click occurred and that the corresponding URI should be activated.

Required	No
Parent	Linear in both Wrapper and InLine formats.
Bounded	0+
Content	A URI for tracking custom interactions.
Attributes	Description
id	A unique ID for the custom click to be tracked.

3.11 Icons

TOC Schema

The industry icon feature was defined in VAST 3.0 to support initiatives such as privacy programs. An example of such a program is the AdChoices program for interest-based advertising (IBA). Though the VAST icon feature was initially created to support privacy programs, it was designed to support other programs that require posting an icon with the linear ad.

This feature is only offered for linear ads because icons can be easily inserted in nonlinear ads and companion creative using existing features. Icon source files may also be included in a wrapper if necessary.

The structure for linear icons uses the `<Icons>` element (plural) as a container for one or more `<Icon>` elements (singular). Each `<Icon>` element provides containers for the creative resource file in section 3.15. Icon tracking is described in sections 0 to 3.11.5.

See section 2.3.8 for details about industry icon support in VAST.

Required	No
Parent	Linear in both InLine and Wrapper formats
Bounded	0-1
Sub-elements	Icon

3.11.1 Icon

TOC Schema

Nested under the `<Icons>` element, the Icon is used to provide one or more creative files for the icon that represents the program being implemented along with any icon tracking elements. Multiple `<Icon>` elements may be used to represent multiple programs.

Required	At least one is required if `<Icons>` is used
Parent	Icons for both InLine and Wrapper formats
Bounded	1+ (if `<Icons>` is used)
Sub-Elements	StaticResource IFrameResource HTMLResource IconClicks IconViewTracking
Attributes	Description
program	The program represented in the icon (e.g. "AdChoices").
width	Pixel width of the icon asset.
height	Pixel height of the icon asset.
xPosition	The x-cooridinate of the top, left corner of the icon asset relative to the ad display area. Values of "left" or "right" also accepted and indicate the leftmost or rightmost available position for the icon asset.
yPosition	The y-cooridinate of the top left corner of the icon asset relative to the ad display area; values of "top" or "bottom" also accepted and indicate the topmost or bottommost available position for the icon asset.
duration	The duration the icon should be displayed unless clicked or ad is finished playing; provided in the format HH:MM:SS.mmm or HH:MM:SS where .mmm is milleseconds and optional.
offset	The time of delay from when the associated linear creative begins playing to when the icon should be displayed; provided in the format HH:MM:SS.mmm or HH:MM:SS.
apiFramework	Identifies the API needed to execute the icon resource file if applicable.
pxratio	The pixel ratio for which the icon creative is intended. The pixel ratio is the ratio of physical pixels on the device to the device-independent pixels. An ad intended for display on a device with a pixel ratio that is twice that of a standard 1:1 pixel ratio would use the value "2." Default value is "1."

3.11.2 IconViewTracking

TOC Schema

The view tracking for icons is used to track when the icon creative is displayed. The player uses the included URI to notify the icon server when the icon has been displayed.

Required	No
Parent	Icon for both InLine and Wrapper formats
Bounded	0+
Content	A URI for the tracking resource file to be called when the icon creative is displayed.

3.11.3 IconClicks

TOC Schema

The `<IconClicks>` element is a container for `<IconClickThrough>` and `<ClickTracking>` elements.

Required	No
Parent	Icon for both InLine and Wrapper formats
Bounded	0-1
Sub-elements	IconClickThrough IconClickTracking

3.11.4 IconClickThrough

TOC Schema

The `<IconClickThrough>` is used to provide a URI to the industry program page that the video player opens when the icon is clicked.

Required	No
Parent	IconClicks for both InLine and Wrapper formats
Bounded	0-1
Content	A URI to the industry program page that is opened when a viewer clicks the icon.

3.11.5 IconClickTracking

TOC Schema

`<IconClickTracking>` is used to track click activity within the icon.

Required	No
Parent	IconClicks for both InLine and Wrapper formats
Bounded	0+
Content	A URI to the tracking resource file to be called when a click corresponding to the `id` attribute (if provided) occurs.
Attributes	Description
id	An id for the click to be measured.

3.12 NonLinearAds

TOC Schema

Nonlinear ads are the overlay ads that display as an image or rich media on top of video content during playback. Within an InLine ad, at least one of `<Linear>` or `<NonLinearAds>` needs to be provided within the `<Creative>` element.

The `<NonLinearAds>` element is a container for the `<NonLinear>` creative files and tracking resources. If used in a wrapper, only the tracking elements are available. Nonlinear creative cannot be provided in a wrapper ad. Nonlinear ad creative use non-video creative files that are describe in section 3.15. Tracking event elements are described in section 0. Ad parameters are used to provide contextual information to the ad and are described in section 3.8.2.

Required	At least one of Linear or NonLinearAds is required.
Parent	Creative for both InLine and Wrapper formats
Bounded	0-1
Sub-elements	NonLinear TrackingEvents

3.12.1 NonLinear

TOC Schema

Each `<NonLinear>` element may provide different versions of the same creative using the `<StaticResource>`, `<IFrameResource>`, and `<HTMLResource>` elements in the InLine VAST response. In a Wrapper response, only tracking elements may be provided.

Required	Yes if `<NonLinearAds>` is used.
Parent	NonLinearAds for both InLine and Wrapper formats
Bounded	1+ (if `<NonLinearAds>` is used)
Sub-elements	StaticResource (InLine only) IFrameResource (InLine only) HTMLResource (InLine only) AdParameters (InLine only) NonLinearClickThrough (InLine only) NonLinearclickTracking (InLine and Wrapper)

3.12.2 NonLinearClickThrough

TOC Schema

Most nonlinear creative can provide a clickthrough of their own, but in the case where the creative cannot provide a clickthrough, such as with a simple static image, the `<NonLinearClickThrough>` element can be used to provide the clickthrough.

A clickthrough may need to be provided for an InLine ad in the following situations:

- Static image file
- Flash file with no API framework
- Flash file in which apiFramework=clickTAG
- Any static resource file where the video player handles the click, such as when "playerHandlesClick=true" in VPAID

`NonLinearClickThrough` is only available for InLine ads.

Required	No
Parent	NonLinear only in InLine format
Bounded	0-1
Content	A URI to the advertiser's page that the video player opens when the viewer clicks the nonlinear ad.

3.12.3 NonLinearClickTracking

TOC Schema

When the nonlinear ad creative handles the clickthrough in an InLine ad, the `<NonLinearClickTracking>` element is used to track the click, provided the ad has a way to notify the player that that ad was clicked, such as when using a VPAID ad unit. The NonLinearClickTracking element is also used to track clicks in wrappers.

NonLinearClickTracking might be used for an inline ad when:

- Any static resource file where the video player handles the click, such as when "playerHandlesClick=true" in VPAID

NonLinearClickTracking is used in a wrapper ad in the following situations:

- Static image file
- Flash file with no API framework
- Flash file in which apiFramework=clickTAG
- Any static resource file where the video player handles the click, such as when "playerHandlesClick=true" in VPAID

Required	No
Parent	NonLinear for both InLine and Wrapper formats
Bounded	0+
Content	A URI to a tracking resource file used to track a nonlinear clickthrough
Attributes	Description
id	An id provided by the ad server to track the click in reports.

3.13 CompanionAds

TOC Schema

Companion ads are display ads that are displayed in the webpage beside the video player or as a skin for the player. The `<CompanionAds>` element is a container for one or more `<Companion>` elements, where each Companion element provides the creative files and tracking details. Companion ads, including any creative, may be included in both InLine and Wrapper formatted VAST ads.

The `required` attribute for the `<CompanionAds>` element provides information about which Companion creative to display when multiple Companions are supplied and whether the Ad can be displayed without its Companion creative. The value for `required` can be one of three values: all, any, or none.

The expected behavior for displaying Companion ads depends on the following values:

- **all:** the video player must attempt to display the contents for all `<Companion>` elements provided. If all companion creative cannot be displayed, the ad should be disregarded and the ad server should be notified using the `<Error>` element.
- **any:** the video player must attempt to display content from at least one of the `<Companion>` elements provided (i.e. display the one with dimensions that best fit the page). If none of the companion creative can be displayed, the ad should be disregarded and the ad server should be notified using the `<Error>` element.
- **none:** the video player may choose to not display any of the companion creative, but is not restricted from doing so. The ad server may use this option when the advertiser prefers that the master linear or nonlinear ad be displayed even if the companion cannot be displayed.

If not provided, the video player can choose to display content from any or none of the `<Companion>` elements. In all cases the video player should display Companion creative at the same time as the linear or nonlinear master creative.

Required	No
Parent	Creative for both InLine and Wrapper formats
Bounded	0-1
Sub-elements	Companion
Attributes	Description
required	Accepts one of the following values: “all” “any” or “none.” See descriptions listed in this section.

3.13.1 Companion

TOC Schema

Both InLine and Wrapper VAST responses may contain multiple companion items where each one may contain one or more creative resource files using the elements: `StaticResource`, `IFrameResource`, and `HTMLResource`. Each `<Companion>` element may provide different versions of the same creative.

The resource elements for providing creative resources are defined in section 3.15. Tracking elements are also available for each companion element. Ad parameters are used to provide contextual information to the ad and are described in section 3.8.2.

Required	At least one Companion is required if `CompanionAds` is used
Parent	CompanionAds for both InLine and Wrapper formats
Bounded	1+ if `<CompanionAds>` is used
Sub-elements	StaticResource IFrameResource HTMLResource AdParameters CcompanionClickThrough CompanionClickTracking
Attributes	Description
width*	The pixel width of the placement slot for which the creative is intended.
height*	The pixel height of the placement slot for which the creative is intended.
id	An optional identifier for the creative.
assetWidth	The pixel width of the creative.
assetHeight	The pixel height of the creative.
expandedWidth	The maximum pixel width of the creative in its expanded state.
expandedHeight	The maximum pixel height of the creative in its expanded state.
apiFramework	The API necessary to communicate with the creative if available.
adslotID	Used to identify desired placement on a publisher's page. Values to be used should be discussed between publishers and advertisers.
pxratio	The pixel ratio for which the companion creative is intended. The pixel ratio is the ratio of physical pixels on the device to the device-independent pixels. An ad intended for display on a device with a pixel ratio that is twice that of a standard 1:1 pixel ratio would use the value "2." Default value is "1."

*required

3.13.2 AltText

TOC Schema

The AltText element is used to provide a description of the companion creative when an ad viewer mouses over the ad.

Required	No
Parent	Companion for both InLine and Wrapper formats
Bounded	0-1
Content	A string to describe the creative when an ad viewer mouses over the ad.

3.13.3 CompanionClickThrough

TOC Schema

Most companion creative can provide a clickthrough of their own, but in the case where the creative cannot provide a clickthrough, such as with a simple static image, the CompanionClickThrough element can be used to provide the clickthrough.

A clickthrough may need to be provided for an InLine ad in the following situations:

- Static image file
- Flash file with no API framework
- Flash file in which apiFramework=clickTAG
- Any static resource file where the video player handles the click, such as when "playerHandlesClick=true" in VPAID

Required	No
Parent	Companion for both InLine and Wrapper formats
Bounded	0-1
Content	A URI to the advertiser's page that the video player opens when the viewer clicks the companion ad.

3.13.4 CompanionClickTracking

TOC Schema

When the companion ad creative handles the clickthrough in an InLine ad, the CompanionClickTracking element is used to track the click, provided the ad has a way to notify the player that that ad was clicked, such as when using a VPAID ad unit. The CompanionClickTracking element is also used in wrappers to track clicks that occur for the Companion creative in the InLine ad that is returned after one or more wrappers.

CompanionClickTracking might be used for an InLine ad when:

- Any static resource file where the video player handles the click, such as when "playerHandlesClick=true" in VPAID

CompanionClickTracking is used in a wrapper ad in the following situations:

- Static image file
- Flash file with no API framework
- Flash file in which apiFramework=clickTAG
- Any static resource file where the video player handles the click, such as when "playerHandlesClick=true" in VPAID
- Any static resource file where the video player handles the click, such as when "playerHandlesClick=true" in VPAID

Required	No
Parent	Companion for both InLine and Wrapper formats
Bounded	0+
Content	A URI to a tracking resource file used to track a companion clickthrough
Attributes	Description
id	An id provided by the ad server to track the click in reports.

3.14 Tracking Event Elements

TOC Schema

The `<TrackingEvents>` element is a container for `<Tracking>` elements used to define specific tracking events described in section 3.14.1. Multiple tracking events can be used to help all the relevant parties track the ad's performance. Each tracking event URI should be included one `<Tracking>` element, using the `event` attribute to identify which event is to be tracked.

The following example shows the section of a VAST response that represents 3 tracking events: two start events, each for a different server, and a complete event.

```
<TrackingEvents>
      <Tracking event="start">
            ![CDATA[http://server1.com/start.jpg]
      </Tracking>
      <Tracking event="start">
            ![CDATA[http://server2.com/start2.jpg]
      </Tracking>
<Tracking event="complete">
            ![CDATA[http://server1.com/complete.jpg]
      </Tracking>
</TrackingEvents>
```

3.14.1 Tracking Event Descriptions

TOC Schema

VAST is used to track a number of ad events using the `<TrackingEvents>` and `<Tracking>` elements. Tracking for impressions is covered in section 3.4.3 and clickthroughs are covered in their relevant sections. Review the schema in section 5 to find more details about tracking the different ad types in VAST. Each `<Tracking>` element contains a URI for the tracking resource of one event. The video player uses these URIs to notify the ad server when the identified event occurs.

In some cases the video player cannot detect that an event has occurred unless the ad communicates the event using a framework such a VPAID. For example, tracking the NonLinear event for `adExpand` requires the ad to notify the video player that it has expanded. This can be done using VPAID or some other framework, but without a means of communication between the ad and the video player, the video player has no way of detecting when the ad has expanded.

The values accepted for tracking events are described in the following list:

Player Operation Metrics (for use in Linear and NonLinear ads)

- **mute:** the user activated the mute control and muted the creative.
- **unmute:** the user activated the mute control and unmuted the creative.
- **pause:** the user clicked the pause control and stopped the creative.
- **resume:** the user activated the resume control after the creative had been stopped or paused.
- **rewind:** the user activated the rewind control to access a previous point in the creative timeline.
- **skip:** the user activated a skip control to skip the creative, which is a different control than the one used to close the creative.
- **progress:** the creative played for a duration at normal speed that is equal to or greater than the value provided in an additional `offset` attribute for the Linear element. Offset values can be time in the format `HH:MM:SS` or `HH:MM:SS.mmm` or a percentage value in the format `n%`. Multiple progress events with different values can be used to track multiple progress points in the Linear creative timeline. This event can be used in addition to or instead of the "quartile" events (firstQuartile, midpoint, thridQuartile, complete). The additional offset attribute can be used to trigger the `creativeView` event to help track a view when an agreed upon duration or percentage of the ad has played.
- **playerExpand:** the user activated a control to extend the player to a larger size.
- **playerCollapse:** the user activated a control to reduce player to a smaller size.

Linear Ad Metrics

- **start:** this event is used to indicate that an individual creative within the ad was loaded and playback began. As with creativeView, this event is another way of tracking creative playback.
- **firstQuartile:** the creative played continuously for at least 25% of the total duration at normal speed.
- **midpoint:** the creative played continuously for at least 50% of the total duration at normal speed.
- **thirdQuartile:** the creative played continuously for at least 75% of the duration at normal speed.
- **complete:** the creative was played to the end at normal speed so that 100% of the creative was played.
- **acceptInvitationLinear:** the user activated a control that launched an additional portion of the linear creative.
- **timeSpentViewing:** amount of video viewed at normal speed in seconds or other appropriate time-based units. If a rewind event occurs during play, time spent viewing may be calculated on total amount of video viewed at normal speed, which may include additional amounts of video viewed after rewinding.
- **otherAdInteraction:** an optional metric that can capture all other user interactions under one metric such a s hover-overs, or custom clicks. It should NOT replace clickthrough events or other existing events like mute, unmute, pause, etc.

NonLinear Ad Metrics

- **creativeView:** not to be confused with an impression, this event indicates that an individual creative portion of the ad was viewed. An impression indicates that at least a portion of the ad was displayed; however an ad may be composed of multiple creative, or creative that only play on some platforms and not others. This event enables ad servers to track which ad creative are viewed, and therefore, which platforms are more common.
- **acceptInvitation:** the user clicked or otherwise activated a control used to pause streaming content, which either expands the ad within the player's viewable area or "takes-over" the streaming content area by launching an additional portion of the ad. An ad in video format ad is usually played upon acceptance, but other forms of media such as games, animation, tutorials, social media, or other engaging media are also used.
- **adExpand:** the user activated a control to expand the creative.
- **adCollapse:** the user activated a control to reduce the creative to its original dimensions.
- **minimize:** the user clicked or otherwise activated a control used to minimize the ad to a size smaller than a collapsed ad but without fully dispatching the ad from the player environment. Unlike a collapsed ad that is big enough to display it's message, the minimized ad is only big enough to offer a control that enables the user to redisplay the ad if desired.
- **close:** the user clicked or otherwise activated a control for removing the ad, which fully dispatches the ad from the player environment in a manner that does not allow the user to re-display the ad.
- **overlayViewDuration:** the time that the initial ad is displayed. This time is based on the time between the impression and either the completed length of display based on the agreement between transactional parties or a close, minimize, or accept invitation event.
- **otherAdInteraction:** an optional metric that can capture all other user interactions under one metric such a s hover-overs, or custom clicks. It should NOT replace clickthrough events or other existing events like mute, unmute, pause, etc.

Companion Ad Metric

- **creativeView:** since companion ads use browser technology for display, tracking metrics can be built into the creative. The only VAST event available for tracking companion creative is the creativeView event. This event enables ad servers to track when companion creative are viewed.

3.14.2 TrackingEvents

TOC Schema

The `<TrackingEvents>` element is available for `Linear`, `NonLinear`, and `Companion`, elements in both InLine and Wrapper formats. When the video player detects that a specified event occurs, the video player is required to trigger the tracking resource URI provided in the nested `<Tracking>` element. When the server receives this request, it records the event and the time it occurred.

Required	No
Parent	Linear NonLinear Companion
Bounded	0-1
Sub-elements	Tracking

3.14.3 Tracking

TOC Schema

Each `<Tracking>` element is used to define a single event to be tracked. Multiple tracking elements may be used to define multiple events to be tracked, but may also be used to track events of the same type for multiple parties.

Required	No
Parent	TrackingEvents for both InLine and Wrapper formats
Bounded	0+
Content	A URI to the tracking resource for the event specified using the `event` attribute.
Attributes	Description
event	A string that defines the event being track. Accepted values are listed in section 3.14.1 and differ for `<Linear>`, `<NonLinear>`, and `<Companion>`.

3.15 Creative Resource Files for Non-Video Creative

TOC Schema

Nonlinear ads, companions, and industry icons are non-video creative, so creative files are nested using elements that define the type of creative resource file provided: `StaticResource`, `IFrameResource`, and `HTMLResource`.

These resource nodes are available under the elements: `<NonLinear>`, `<Companion>`, and `<Icon>` in the inline format; however, in wrapper format, resource files may only be provided under the `<Companion>` and `<Icon>` elements. Nonlinear elements in wrapper format are only used for tracking, and resource files are not allowed.

Multiple creative files may be included using these components, but each element should contain one or more files to represent different versions of the creative for use in different environments. The video player can choose which file to use when more than one resource file is provided within a single container.

For example, if an ad server wants to submit both a static image and an HTML creative for a nonlinear ad, then the nonlinear portion of the VAST response would be formatted as follows:

```
<NonLinearAds>
      <NonLinear>
            <StaticResource>
                  ![CDATA[http://adserver.com/staticresourcefile.jpg]
            </StaticResource>
            <HTMLResource>
                  ![CDATA[http://adserver.com/htmlresourcefile.htm>
            </HTMLResource>
      </NonLinear>
</NonLinearAds>
```

The three resource file elements are described in the following sections.

3.15.1 StaticResource

TOC Schema

The URI to a static creative file to be used for the ad component identified in the parent element, which is either: <NonLinear>, <Companion>, or <Icon>.

Required	One of <StaticResource>, <IFrameResource>, or <HTMLResource> is required if <NonLinear>, <Companion>, or <Icon> is used
Parent	NonLinear, Companion, or Icon in the InLine format Companion or Icon in the Wrapper format (Resource files are not provided for nonlinear ads in a wrapper)
Bounded	0+
Content	A URI to the static creative file to be used for the ad component identified in the parent element.
Attributes	Description
creativeType*	Identifies the MIME type of the creative provided.

*required

3.15.2 IFrameResource

TOC Schema

The URI to a static creative file to be used for the ad component identified in the parent element, which is either: <NonLinear>, <Companion>, or <Icon>.

Required	One of <StaticResource>, <IFrameResource>, or <HTMLResource> is required if <NonLinear>, <Companion>, or <Icon> is used
Parent	NonLinear, Companion, or Icon in the InLine format Companion or Icon in the Wrapper format (Resource files are not provided for nonlinear ads in a wrapper)
Bounded	0+
Content	A URI to the iframe creative file to be used for the ad component identified in the parent element.

3.15.3 HTMLResource

TOC Schema

The URI to a static creative file to be used for the ad component identified in the parent element, which is either: `<NonLinear>`, `<Companion>`, or `<Icon>`.

Required	One of `<StaticResource>`, `<IFrameResource>`, or `<HTMLResource>` is required if `<NonLinear>`, `<Companion>`, or `<Icon>` is used in the Inline format
Parent	NonLinear, Companion, or Icon in the InLine format Companion or Icon in the Wrapper format (Resource files are not provided for nonlinear ads in a wrapper)
Bounded	0+
Content	A URI to the static creative file to be used for the ad component identified in the parent element.

3.16 AdVerifications

TOC Schema

The `<AdVerifications>` element is used to contain one or more `<Verification>` elements, which are used to initiate a controlled container where code can be executed for collecting data to verify ad playback details.

If VPAID is being used for ad verification, the ad verification VPAID unit should be included under the `<Verification>` element as either a `<JavaScriptResource>` or `<FlashResource>`.

Required	No
Parent	InLine
Bounded	0-1
Sub-elements	Verification

3.17 Verification

TOC Schema

Nested under `AdVerifications`, the `Verification` element is used to contain the JavaScript or Flash code used to collect data. Multiple Verification elements may be used in cases where more than one verification vendor needs to collect data or when different API frameworks are used. Each `<Verification>` element should contain one of either the `<JavaScriptResource>` or the `<FlashResource>`. If one verification vendor is providing code in both Flash and JavaScript as an option for the player, more than one of the resource elements may be included.

When included, verification contents must be executed (if possible) BEFORE the media file or interactive creative file is executed, to ensure verification can track ad play as intended. Players should attempt to execute the provided file or use the `<Error>` URI, if provided, to notify the ad server. Error code 410 can substitute the URI macro, if one was provided.

Required	No
Parent	AdVerifications
Bounded	0+
Sub-elements	JavaScriptResource FlasResource ViewableImpression
Attributes	Description
vendor	The home page URL for the verification service provider that supplies the resource file.

3.17.1 JavaScriptResource

TOC Schema

A container for the URI to the JavaScript file used to collect verification data.

Required	No
Parent	Verification
Bounded	0+
Content	A CDATA-wrapped URI to the JavaScript used to collect data
Attributes	Description
APIFramework	The name of the API framework used to execute the AdVerification code

3.17.2 FlashResource

TOC Schema

A container for the URI to the Flash file used to collect verification data.

Required	No
Parent	Verification
Bounded	0+
Content	A CDATA-wrapped URI to the Flash used to collect data
Attributes	Description
APIFramework	The name of the API framework used to execute the `AdVerification` code

3.17.3 ViewableImpression

TOC Schema

The verification vendor may provide URIs for tracking viewable impressions under both the InLine ad and any Wrappers using the `<ViewableImpression>` element. The player is not required to do anything with the viewable impression element for verification.

A publisher may choose to use the `<ViewableImpression>` element for the `<Ad>` described in section 3.5.

Required	No
Parent	Verification
Bounded	0-1
Contents	A URI to the tracking resource file for the verification viewable impression
Attributes	Description

id	An ad server id for the viewable impression.

3.18 Extensions

TOC Schema

Ad servers can use this XML node for custom extensions of VAST. When used, custom XML should fall under the nested `<Extension>` (singular) element so that custom XML can be separated from VAST elements. An XML namespace (xmlns) should also be used for the custom extension to separate it from VAST components.

The following example includes a custom xml element within the `<Extensions>` element.

```
<Extensions>
    <Extension>
    <CustomXML>…</CustomXML>
    <Extension>
</Extensions>
```

The publisher must be aware of and be capable of executing any VAST extensions in order to process the content.

Required	No
Parent	InLine or Wrapper
Bounded	0-1
Sub-elements	Extension

3.18.1 Extension

TOC Schema

One instance of `<Extension>` should be used for each custom extension. The `type` attribute identifies the MIME type of any code provided in the extension.

Required	No
Parent	Extensions
Bounded	0+
Content	Custom XML object
Attributes	Description
type	The MIME type of any code that might be included in the extension.

3.19 Wrapper

TOC Schema

VAST wrappers are used to redirect the video player to another server for either an additional `<Wrapper>` or the VAST `<InLine>` ad. In addition to the URI that points to another file, the wrapper may contain tracking elements that provide tracking for the inline ad that is served following one or more wrappers. A wrapper may also contain `<Companion>` creative and `<Icon>` creative. And while `<Linear>` and `<NonLinear>` elements are available in the wrapper, they are only used for tracking. No media files are provided for Linear elements, nor are resource files provided for NonLinear elements. Other elements offered for InLine ads may not be offered for wrappers.

To find out if an element is offered for wrappers, check the human-readable schema in section 5.

Required	One of either InLine or Wrapper required but both are not allowed
Parent	Ad
Bounded	0-1
Sub-elements	Impression* VASTAdTagURI* AdSystem Error ViewableImpression AdVerifications Extensions Creatives
Attributes	Description
followAddtionalWrappers	a Boolean value that identifies whether subsequent wrappers after a requested VAST response is allowed. If false, any Wrappers received (i.e. not an Inline VAST response) should be ignored. Otherwise, VAST Wrappers received should be accepted (default value is "true.")
allowMultipleAds	a Boolean value that identifies whether multiple ads are allowed in the requested VAST response. If true, both Pods and stand-alone ads are allowed. If false, only the first stand-alone Ad (i.e. no sequence value for the Ad) in the requested VAST response is allowed. Default value is "false."
fallbackOnNoAd	a Boolean value that provides instruction for using an available Ad when the requested VAST response returns no ads. If true, the video player should select from any stand-alone ads available. If false and the Wrapper represents an Ad in a Pod, the video player should move on to the next Ad in a Pod; otherwise, the video player can follow through at its own discretion where no-ad responses are concerned.

3.19.1 VASTAdTagURI

TOC Schema

While VAST wrappers don't provide all the same elements offered for an inline ad, the `<VASTAdTagURI>` is the only element that is unique to wrappers. The `VASTAdTagURI` is used to provide a URI to a secondary VAST response. This secondary response may be another wrapper, but eventually a VAST wrapper must return an `<InLine>` ad. In VAST 4.0 the player is only required to accept five wrappers ads. If no inline ads are returned after 5 wrappers, the player may move on to the next option.

Required	Yes (if `<Wrapper>` is used)
Parent	Wrapper
Bounded	0-1
Content	A URI to another VAST response that may be another VAST Wrapper or a VAST InLine ad. The number of subsequent VAST responses that include wrappers should not exceed five before an inline ad is served. After five VAST wrapper responses, acceptance of additional VAST responses is at the publisher's discretion.

4 Migration to VAST 4.0

VAST 4.0 offers features to support long-form video, server-side tracking, industry-wide creative tracking, and viewability and verification tracking. While the advance in features is alluring, video players will need time to upgrade their systems. During the transition period from VAST 3.0 to 4.0 (or 2.0 to 4.0), prepare to manage varying feature support in the market. VAST 4.0 was designed to be backwards compatible with version 3.0 and VAST 3.0 was designed to be backwards compatible with version 2.0. However, features introduced in the newer versions will not be supported in older-versioned players. The following sections outline a few notes to consider as VAST 4.0 is introduced into the market.

4.1 Advertisers and Ad Technology Vendors

Design ads that can be successfully delivered to lower versioned VAST players while still optimizing the response with new 4.0 capabilities. For example:

- VAST 4.0 ads discourage the use of VPAID or other interactive ad units that require an API to execute. The new `<InteractiveCreativeFile>` was provided to accommodate such ads. However, in older versions, an interactive unit may be provided in addition to the video `<MediaFile>` in order to ensure interactive files are executed where possible in older VAST version players.
- In a 4.0 response, use both the Creative `adId` attribute as well as the new `<UniversalAdId>` element to provide a creative ad ID.

4.2 Ad Servers and Networks

Be prepared to manage the variability with VAST versions. For example, if a player specifically requests a VAST 3.0 response, then the ad server should limit responses to VAST 3.0.

If one or more verification vendors are involved, use VAST 4.0 to provide verification code in the new `<AdVerification>` node, but expect that older versioned players will not recognize the verification node. If VPAID is being used for verification, consider alternate methods of supplying the VPAID unit to the player in older versions in order to prevent latency.

4.3 Video Players

VAST 4.0 players should continue to accept VAST 3.0 and VAST 2.0 ads. If there are no plans to upgrade to 4.0, test whether 4.0 ads can play in your player at the 2.0 or 3.0 version that is supported.

5 Human Readable VAST XML Schema

The following schema models the structure for VAST along with available attributes. Click the section number for more detail.

Element	Attributes	Required	Section
VAST	version	Yes	0
/Error		No	3.2.1
VAST/Ad	id, sequence	Yes	3.3
VAST/Ad/InLine		Yes*	0
/AdSystem	version	Yes	3.4.1
/AdTitle		Yes	3.4.2
/Impression	id	Yes	3.4.3
/Category	authority	Yes	3.4.3
/Description		No	3.4.5
/Advertiser		No	3.4.6
/Pricing	model, currency	No	3.4.7
/Survey		No	0
/Error		No	3.4.9
/ViewableImpression	id	No	3.5
/Viewable		No	3.5.1
/NotViewable		No	3.5.2
/ViewUndetermined		No	0
/AdVerifications		No	3.16
/Verification	vendor	No	3.17
/JavaScriptResource	apiFramework	No	3.17.1
/FlashResource	apiFramework	No	3.17.2
/ViewableImpression	id	No	3.17.3
/Extensions		No	3.17.3
/Extension	type	Yes	3.18.1
/Creatives		Yes	3.6
/Creative	id, sequence, adId, apiFramework	Yes	0
/UniversalAdId	idRegistry, idValue	Yes	3.7.1
/CreativeExtensions		No	3.7.2
/CreativeExtension			3.7.3
/Linear	Skipoffset	Yes (linear)	3.8
/Duration		Yes	3.8.1
/AdParameters	xmlEncoded	No	3.8.2
/MediaFiles		Yes	3.9
/Mezzanine		Yes (ad-stitching)	3.9.2

/MediaFile	id, delivery, type, bitrate, minBitrate, maxBitrate, width, height, scalable, mantainAspectRatio, codec, apiFramework	Yes	3.9.1
/InteractiveCreativeFile		No	3.9.3
/VideoClicks		No	3.10
/ClickThrough	id	No	3.10.1
/ClickTracking	id	No	0
/CustomClick	id	No	3.10.3
/TrackingEvents		No	0
/Tracking	event, offset	No	3.14.3
/Icons		No	3.11
/Icon	program, width, height, xPosition, yPosition duration, offset, apiFramework, pxratio	Yes	3.11.1
/StaticResource /IFrameResource /HTMLResource	creativeType (StaticResource only)	Yes	3.15.1
/IconClicks		No	3.11.3
/IconClickThrough		No	3.11.4
/IconClickTracking	id	No	0
/IconViewTracking		No	0
/NonLinearAds		Yes (nonlinear ads)	0
/NonLinear			3.12.1
/NonLinearClickThrough			3.12.2
/NonLinearClickTracking			3.12.2
/CompanionAds	Required	No	0
/Companion	id, width, height, assetWidth, assetHeight, expandedWidth, expandedHeight, apiFramework, adSlotID, logoTile, logoTitle, logoArtist, logoURL, pxratio	No	3.13.1
/StaticResource /IFrameResource /HTMLResource	creativeType (StaticResource only)	Yes	3.15.1
/AdParameters	xmlEncoded	No	3.8.2
/AltText		No	3.13.2
/CompanionClickThrough		No	3.13.3
/CompanionClickTracking	id	No	3.13.4

VAST/Ad/Wrapper	followAdditonalWrappers, allowMultipleAds, fallbackOnNoAd	No*	3.19
/Impression	id	Yes	3.4.3
/VASTAdTagURI		Yes	3.19.1
/AdSystem	version	No	3.4.1
/Pricing	model, currency	No	3.4.7
/Error		No	3.4.9
/ViewableImpression	id	No	3.5
/Viewable		No	3.5.1
/NotViewable		No	3.5.2
/ViewUndetermined		No	0
/AdVerifications		No	3.16
/Verification	vendor	No	3.17
/ViewableImpression	id	No	3.17.3
/Extensions		No	3.18
/Extension	type	No	3.18.1
/Creatives		No	3.6
/Creative	id, sequence, adId	No	0
/Linear		Yes	3.8
/TrackingEvents		No	0
/Tracking	event, offset	No	3.14.3
/VideoClicks		No	3.10
/ClickTracking	id	No	0
/CustomClick	id	No	3.10.3
/Icons		No	3.11
/Icon	program, width, height, xPosition, yPosition duration, offset, apiFramework, pxratio	Yes	3.11.1
/StaticResource /IFrameResource /HTMLResource	creativeType (StaticResource only)	No	3.15
/IconClicks		No	3.11.3
/IconClickThrough		No	3.11.4
/IconClickTracking		No	3.11.5
/IconViewTracking		No	0
/InteractiveCreativeFile			3.9.3
/CompanionAds	required	No	0
/Companion	id, width, height, assetWidth, assetHeight, expandedWidth, expandedHeight, apiFramework, adSlotID, logoTile, logoTitle, logoArtist, logoURL, pxratio	No	3.13.1

/StaticResource /IFrameResource /HTMLResource	creativeType (StaticResource only)	No	3.15
/AdParameters	xmlEncoded	No	3.8.2
/AltText		No	3.13.2
/CompanionClickThrough		No	3.13.3
/CompanionClickTracking		No	3.13.4
/TrackingEvents		No	0
/Tracking	event	No	3.14.3

*Either the InLine element or the Wrapper element is required and only one is allowed.

6 VAST Terminology

As the video advertising industry has evolved, certain terminology has gained widespread adoption. The following definitions represent some of that terminology as it relates to video ad serving discussed in this document.

Ad Pod: An ad Pod is sequence of Linear ads played back-to-back, like a commercial break with multiple ad spots on TV.

Companion Ad: Commonly a display banner or rich media ad that appears on the page outside of the video player. Companion ads may remain on the page after the related in-stream ad ends. A Companion ad can also be a skin that wraps the video experience.

Clickthrough: A URL for page that opens when a user clicks the ad creative.

InLine Ad: A VAST ad response that contains all the information needed to display the video ad. No additional calls to other ad servers are needed after a VAST InLine ad response is received.

In-Stream Ad: Any ad that appears inside a streaming video player, whether it's an image overlay or a Linear video ad, such as an ad that plays in a 30 second ad spot.

Linear Ad: Linear ads are like TV commercials and can appear before the content video plays (pre-roll), during a break in the content video (mid-roll), or after the content video ends(post-roll). Linear ads may be video, rich media or still image ads. Using an API or other technology, Linear ads can be interactive and ad duration can be extended when a user interacts.

Master Ad: For video ad campaigns that include an in-stream ad plus one or more Companion ads, the in-stream portion of the ad unit is referred to as the master ad. In this master-companion relationship, the master ad must always be shown.

Nonlinear Ad: An in-stream ad that appears concurrently with the video content playback. Nonlinear ads usually cover the bottom or top fifth of the video player and can be text, image or interactive ads. Using an API or other technology, the video player may allow user-initiated interaction in a nonlinear ad to stop content video playback. Nonlinear ads can

only appear at some point between content video start and end (mid-roll positions) and generally disappear after 10-20 seconds if there is no interaction.

Overlay Ad: A nonlinear ad format in which an image or text displays on top of video content. Overlay ads are commonly referred to as simply “nonlinear ads;” however nonlinear ads may also include non-overlay formats that are served within the video player but without covering any video content.

Primary Ad Server: The first ad server that the video player calls to for ad content. The primary ad server is usually the ad server used by the publisher.

Secondary Ad Server: The ad server that the video player calls after receiving a VAST redirect (wrapper ad) from the primary ad server. Secondary ad servers may include agency or ad network ad servers. Also, secondary ad servers may redirect the video player to a third ad server and the third ad server may redirect to a fourth, and so on. Eventually, an ad server must provide a VAST response that includes all the creative elements needed to display the ad.

VAMG: Video Ad Measurement Guidelines is an IAB guideline that defines the set of events that should be tracked when a video ad is played.

VAST: The Video Ad Serving Template is an IAB guideline and XML schema that describes the XML structure for a video ad response. VAST enables ad responses to come from any ad server.

VAST Redirect: A VAST ad response that points to another VAST response (sometimes referred to as the downstream VAST response).

VAST Tag: A URI that returns a VAST response when called.

Video Ad: Any ad displayed in the context of a video experience. A video experience may include in-banner video, in-text video, in-stream video and other formats. VAST applies only to in-stream video where a video player is used to manage the video experience independent of any other content. For example, video served within an ad banner is considered rich media and is NOT addressed in the VAST guideline.

Video Player: A video playback environment used to manage a video experience. Video players are provided by an Online Video Platform (OVP) vendor or can be custom-built by the publisher.

VMAP: Video Multi Ads Playlist is an IAB guideline that describes the XML structure for a playlist of video ads sent from an ad server to a video player.

VPAID: Video Player Ad Interface Definition is an IAB guideline that defines the communication protocols between an interactive ad and the video player that is rendering it.

Wrapper: in the context of VAST, a Wrapper is a response that provides a URI that the video player uses to call a secondary VAST response. The secondary response may be either another Wrapper or a VAST InLine response.