Selling Video Traffic via Waterfall-Model

When selling Video inventory, increasing the fill rate (viewed content videos vs. paid ad videos) is essential. Unfortunately for publishers the design of VAST and VPAID enables advertisers to omit the delivery of a video ad. Hence publishers cannot rely on displaying one advertiser's VAST file only but must implement structures which allows to try and display alternative advertisers: The so called Waterfall-Model.

What is VAST?

VAST is a (document-) standard which defines a way how advertisers can transmit information about video ads to a player. The VAST-URL or file itself is basically a text-file including multiple elements of information. The publishers video player will download the VAST file, parse its content and thereof know if and what to display as a video ad.

It is important to know, that VAST allows basically three kinds of responses for the advertiser:

a) The VAST file includes information about the video files (mp4, flv, webm or similar) which can then be displayed by the player.
b) The VAST file includes information about VPAID files which can be displayed by the player.
c) The VAST file includes a wrapper tag. This is nothing else than a link to another VAST-file which the player shall call in order to get further information.
d) The VAST file can be empty (no information/ad to display)

What is VPAID?

VPAID is a standard which defines a way how a publisher's video player can integrate another (advertiser's) video player. The goal of this is to allow advertisers to use a player that supports features the advertiser needs (e.g. a certain player design, counting mechanism or other features). In case the VAST document points to a VPAID file, the publishers player will load the advertisers player and from then on the advertisers player is responsible for playing the video ad. Once the advertiser's player is done, the publisher's player takes over again and plays the content video.

What's the problem?

The problem with both standards, VAST and VPAID, is that they allow the advertiser to control whether or not a video ad shall be displayed:

a) An advertiser's adserver can decide to respond with an empty VAST file for a certain user/request
b) An advertiser's VPAID can actively decide not to show an ad.

Also the implementation of the VPAID "protocol" is not very easy, which can lead to problems and errors when a player tries to display an advertiser's VPAID player. The result is that a possible video impression is left empty through a decision/error on advertisers side: The publisher loses money although there might be another advertiser which would have shown an ad on this impression.

What's the solution?

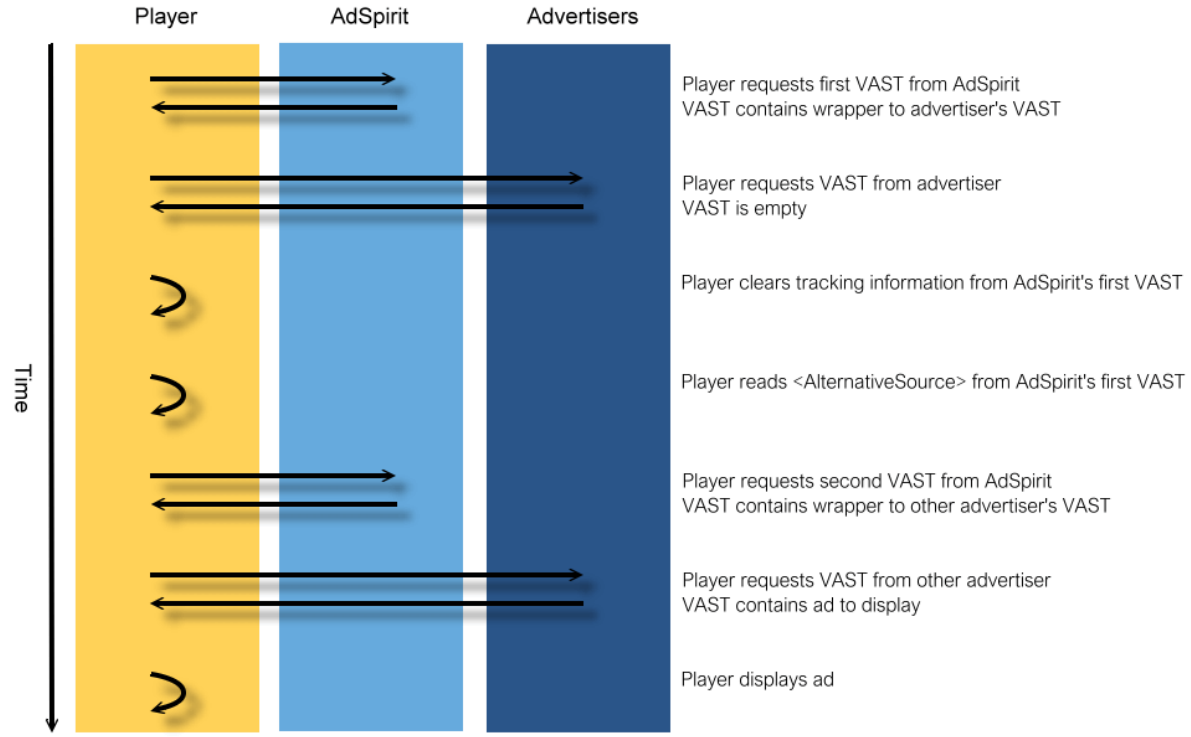
The solution is to implement a so called waterfall-model:
The player will test the first advertiser and its VAST/VPAID. In case there is no ad to display, the player will test the second advertiser and its VAST/VPAID. This is repeated until either an advertiser with an active video ad is found or until the player decides not to test any more advertisers (e.g. tests took too long time and the user shall no longer wait).

Unfortunately VAST does not define a standard way how to tell the publisher's player how/where to look for alternative VAST sources. But VAST allows adserver vendors to create own extensions in order to enrich the VAST file with further information. AdSpirit does this by adding the extension <AlternativeSource> to the VAST file. This will include a VAST-URL to the AdSpirit AdServer which can respond with a different VAST file with information about another campaign/advertiser.

Implementation on publisher's side

In order to use the waterfall model, the publisher's player need to be aligned to AdSpirit's extension of VAST. The extension can be found in the VAST file as a sub-element of <Linear> or <Wrapper>. If the player discovers that the first advertiser's VAST/VPAID is not displayed, it must clear all tracking information from VAST files that were loaded before and restart the VAST process again with the URL found in the Extension.

Example workflow:



Recommendation:
It is also recommended to separate different VAST-URLs into different campaigns rather than one campaign with multiple creatives. This way, the VAST called via AlternativeSource can retrieve the next campaign in row.

What else can be done?

In addition and/or as a replacement to implementing AdSpirit's <AlternativeSource> extension, publishers can use AdSpirit's serverside prefetch. You can enable this by setting "Prefetch VAST from adserver" to "yes" in creative settings. Once this feature is enabled, AdSpirit will download the VAST file from the advertiser before sending the response to the player. Depending on the content of the advertisers VAST file AdSpirit will react as follows:

a) If the advertiser's VAST file contains a video source, AdSpirit will give this video source directly to the publisher.
b) If the advertiser's VAST file contains a VAST wrapper, AdSpirit will download this next VAST file (up to 5 wrappers in a row) and react accordingly.
c) If the advertiser's VAST file contains a VPAID, AdSpirit will give this VPAID directly to the publisher.
d) If the advertiser's VAST file is empty, AdSpirit will give a VAST wrapper to the publisher which leads to AdSpirits VAST file with the current campaign excluded (this way the publishers player will ask AdSpirit again for the next campaign).

As a result AdSpirit will “check” if the advertiser’s VAST is empty or not and will only respond with non-empty VAST files (unless no advertiser remains).

Please note, that AdSpirit will not try to fetch VPAID files (see Point c above). This means the serverside prefetch can only be a workaround for empty VAST wrappers but not for VPAID files that decide not to display an ad.