CTV-OTT PMC: Ad Server Integration with Prebid Server Architecture and Interface Proposal

Contributors: Alex Krassel, Ingrmar Zach, Scott Kay, Bret Gorsline

Last Updated: Aug 23, 2023

Table of Content:

Overview

In the Prebid for CTV-OTT: Use Cases, Common Integration Architectures, Challenges, Solutions document we've outlined the known challenges of typical Prebid Server integration for an SSAI based solution for long form video ad insertion for CTV.

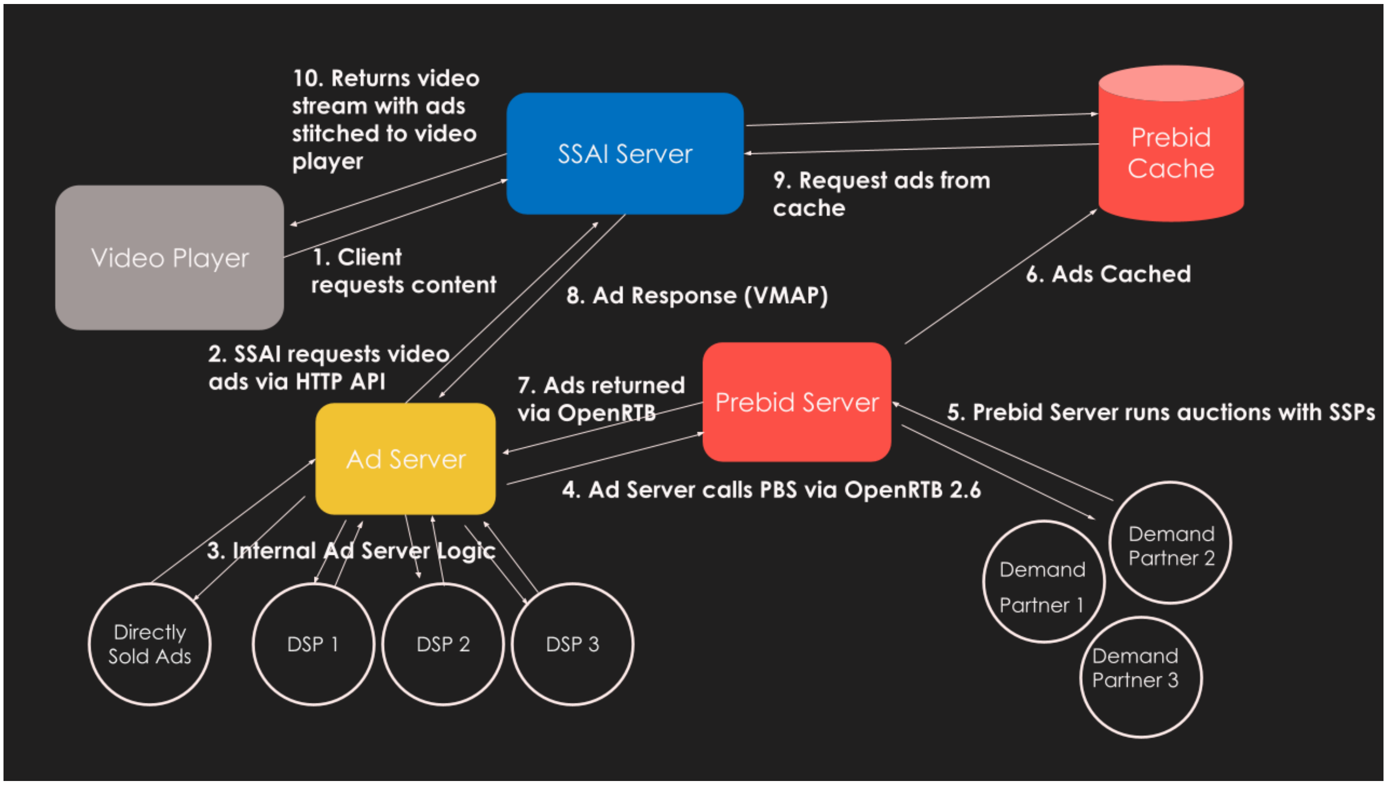
The 2 largest challenges are additional SSAI integration required as well as a large number of key-value pairs based line items that need to be created to accommodate the Prebid solution.

We propose a solution that solves both of these problems by making the ad server explicitly aware and working directly with the Prebid Server.

This proposed solution is still work in progress, and industry feedback is highly welcome to make sure the proposed architecture doesn't miss any significant nuances.

Proposed Architecture

The following diagram depicts the proposed integration architecture:


10. Returns video stream with ads stitched to video player
SSAI Server
Prebid Cache
9. Request ads from cache
Video Player
1. Client requests content
8. Ad Response (VMAP)
6. Ads Cached
2. SSAI requests video ads via HTTP API
7. Ads returned via OpenRTB
Prebid Server
5. Prebid Server runs auctions with SSPs
Ad Server
4. Ad Server calls PBS via OpenRTB 2.6
3. Internal Ad Server Logic
Directly Sold Ads
DSP 1
DSP 2
DSP 3
Demand Partner 1
Demand Partner 2
Demand Partner 3

The integration in this use case works as follows:

1. A viewer using a client-side device (smart TV, Roku box, Fire TV device, Apple TV device, web browser running on a PC, tablet, or mobile phone, etc.) selects a video to watch or tunes into a live streaming channel. This results in the client side video player app making a request to the SSAI server for requested content.
   The aspects of the overall architecture that cover the content retrieval by the SSAI server is beyond the scope of this document, and are omitted in this diagram, as we only cover the ad insertion portion.

2. The SSAI Server only needs to be aware of the Ad Server in this architecture, so it calls the Ad Server as usual.

3. The Ad Server implements its custom logic to collect appropriate directly sold ads as well as programmatic ads from DSPs with which it has established connections.

4. The Ad Server also calls the Prebid Server instance of Publisher's choice via OpenRTB 2.4+ (OpenRTB 2.6 preferred). The Prebid Server instance of Publisher's choice will use parts of the OpenRTB spec that allow requesting multiple ads in one call to improve

monetization and efficiency (if provided). In theory it is possible for the Ad Server to call different instances of the PBS for different publishers if desired.

The call to the Prebid Server can happen in one of the following two approaches:

a. At the same time as step 3 above - which can reduce overall latency.

b. After step 3 - which can improve monetization with a slightly higher latency as a trade off. In this case the Ad Server may be able to start building an ad pod (or multiple pods), so it may have a better understanding on how many Prebid ads it might need, and also what categories may need to be explicitly excluded from Prebid Server's demand sources because these categories have already been selected for the directly sold ads.

5. The Prebid Server runs auctions in parallel against all preconfigured demand sources using appropriate demand source adapters. If the Ad Server passed categories to exclude, these categories are in turn passed to the demand source adapters.
   The Prebid Server also performs necessary deduplication logic on demand adapter responses (most commonly competitive category deduplication).

6. The Prebid Server places all bids / VAST XML tags that survive deduplication in the Prebid Cache.

7. The Prebid Server then returns an OpenRTB 2.6 response back to the Ad Server. All references to the VAST tags returned by the Prebid Server are the URLs of the VAST tags in the Prebid Cache.

8. The Ad Server uses internal logic to combine the directly sold ads, programmatic ads from DSPs as well as the ads returned in the OpenRTB 2.6 format by the Prebid Server into the final VMAP response that includes all the ads for a pod (or all the pods). The Ad Server is responsible for the final deduplication between directly sold and programmatic ads as it constructs the ad pods.

9. The SSAI server processes such VMAP response, and performs the stitching of the ad pod(s) with the main video content. For ads coming from the Prebid Server, the SSAI server retrieves the VAST from the Prebid Cache.

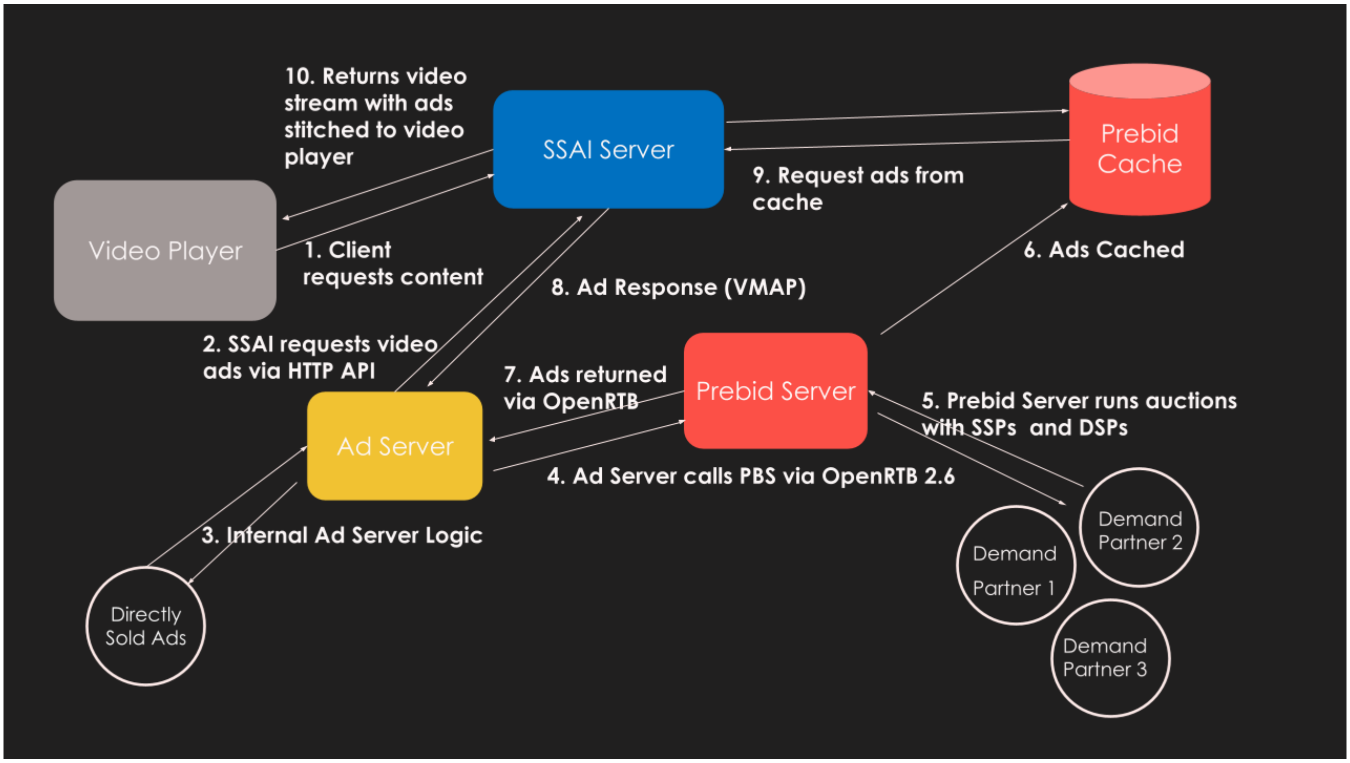
10. The combined stream plays on the viewer's device.

Note: For a single ad pod the Ad Server may return VAST with multiple sequence numbers instead of VMAP. Whether VMAP or VAST is returned to the SSAI Server, it has no bearing on the above architecture.

In the integration architecture proposed above it is theoretically possible to call a particular DSP twice - once directly by the Ad Server, and then again via an appropriate PBS demand adapter.

We advise against it, since it would inadvertently inflate the perception of supply availability, which can lead to lower CPMs. Also, if a demand source is accessed via two separate paths for the same impressions, it would most likely be eventually SPO'd/DPO'd. Our recommendation in this use case is to call such DSP directly, and then make sure PBS is configured not to call this DSP via demand adapters.

Alternatively, the Ad Server can forgo calling any of the programmatic demand sources outside of the Prebid Server directly and completely rely on the Prebid Server's demand adapters that can work with both SSPs and DSPs. The diagram below illustrates the use case where all external demand sources are routed through the Prebid Server:


10. Returns video stream with ads stitched to video player
SSAI Server
Prebid Cache
Video Player
9. Request ads from cache
1. Client requests content
6. Ads Cached
8. Ad Response (VMAP)
2. SSAI requests video ads via HTTP API
7. Ads returned via OpenRTB
Prebid Server
5. Prebid Server runs auctions with SSPs and DSPs
Ad Server
4. Ad Server calls PBS via OpenRTB 2.6
3. Internal Ad Server Logic
Demand Partner 2
Demand Partner 1
Directly Sold Ads
Demand Partner 3

Here Demand Partner 1 - N can be either SSPs or DSPs that provide demand adapters for PBS that support long form video format.

Another advantage of this approach is that the Prebid Server would perform deduplication against all programmatic demand sources, thus making it unnecessary to do that in the Ad Server.

The disadvantage of this approach is that the direct calls to DSPs may fare better from SPO (supply path optimization) perspective.

Theoretically, in either of these architectures, the Ad Server may call multiple Prebid Server instances from different vendors in parallel, but this will most likely not help to improve yield. Ultimately each Prebid Server instance will utilize the same demand adapters, so the calls may look pretty much indistinguishable to the demand sources. Additionally, this may make it look like there is twice as much inventory available to bidders, which will drive CPMs down. And SPO methodology used by the demand sources may end up favoring a path through one PBS instance over the other making the other instance irrelevant.

It is not recommended that multiple Prebid Server be used because the Ad Server would have to perform additional deduplication between ads returned by different PBS instances (since they are not aware of each other).

Additional Considerations

Configuration ID

When the Ad Server calls the Prebid Server, PBS needs to know which adapters need to be invoked and with which parameters. This configuration is provided by the Publisher, since each publisher maintains relationships with specific demand sources. Publishers may choose to include adapter configuration in each request or reference information saved in PBS using the Stored Request feature.

Developing an interface to provide Publishers with a way to configure adapters is a complex task that may require a large investment. To expedite the development process, a quicker solution could be to utilize the PBS Stored Requests feature.

To support such configurability, the Ad Server will have to be able to provide functionality to maintain such Config IDs for Publishers, and to pass the strings to PBS via OpenRTB. It's possible that a server-side module could construct Config IDs from fragments sent by the ad server including ad slot information, browser, etc.

Ad Categories

Ad Categories are important for competitive separation of ads, but most ad servers have their own proprietary definitions of ad categories. Additionally, many ad servers allow publishers to define their own categories.

OpenRTB expects standard IAB ad categories in requests and responses, so an Ad Server <-> IAB category translation needs to be performed for the request as well as the response.

To perform such translation there should be a way to let publishers define their specific category translation rules.

While it might be easier to perform such category translation in a centralized manner in the Prebid Server, the Ad Server may need to implement the translation logic outside of the Prebid Server when it communicates directly with the DSPs via OpenRTB, so it may be unnecessary to implement category translation inside PBS. This seems to also be the opinion of the IAB.

VAST Tag Unwrapping

It may be beneficial for the Prebid Server to perform unwrapping of the VAST tags it returns. Here are the advantages of doing so:

- This may reduce the overall latency because PBS may be unwrapping programmatic tags in parallel with the Ad Server unwrapping direct tags and also tags retrieved separately directly from the DSPs.

- VAST tags may contain additional competitive category information that can also be used for deduplication. (Having stated that, it is worth mentioning that the ad description part of VAST is optional, and the majority of VAST tags obtained through programmatic channels don't seem to have it.)

The consensus of the CTV Taskforce is that VAST unwrapping by the core Prebid Server adds very little value for this use case. As such, it is not required / recommended. The latency reduction makes most sense when PBS is called first before the Ad Server, so it can perform unwrapping while the Ad Server makes the ad serving decisions. Since for this specific use case Ad Server is called first, and then it calls PBS, having PBS unwrap VAST will not significantly affect latency.

VAST unwrapping may serve a much larger role for the use case where the Ad Server is not being used at all, and PBS is called directly to deliver ad pods. This use case, while worth mentioning, lies outside the scope of this architecture proposal document. In this case VAST tags need to be unwrapped by the PBS so the ad duration information could be used by the pod construction module. This functionality would need to be implemented in an optional vendor specific podding module, and not in the core PBS.

Identity

No specific considerations are needed for handling identity within this proposal. Publishers can utilize standard OpenRTB identity frameworks to transmit the required information to Prebid Server, and subsequently to adapters.

Having said this, in the CTV use case the publishers have the 1st party data about the viewers since most CTV apps require viewer login and profile selection. This information can be relayed via OpenRTB to the demand sources for viewers that opt in to do so. Additionally, long form video providers usually have significant amounts of contextual non viewer specific information that can be used for contextual targeting.

In the absence of cookies or MAIDs on most CTV devices, there's no universal privacy-safe match key available for targeting or frequency capping. Publishers and programers want to protect their 1P data and we're looking for industry players to propose a potential extension framework to Prebid-Server enabling 3P vendors to integrate their solutions via a sidecar standard.

More discussion and ideas to follow…

Reporting by Demand Source

Since the Prebid Server consolidates demand from multiple demand sources, it may be important for publishers to know how each of these disparate demand sources perform for them. As a result, the ability to report by demand source may become very important.

A recommended solution for this should utilize the demand source information returned by PBS in the OpenRTB responses as they are being processed. This mechanism can provide multiple pieces of information, including for both bids and non-bids.

Alternatively Prebid Server's Analytics Adapters could be used to obtain the same information. This is something the Ad Server may choose instead of using information from OpenRTB responses.

Dynamic Pods and Ad Duration

In this proposed integration architecture the Ad Server is responsible for constructing pods - the Prebid Server simply returns multiple deduplicated ads in response to the Ad Server OpenRTB requests for multiple ads. As such, in this architecture, PBS does not need to make any dynamic podding decisions based on ad durations or any other information.