mpx: A SUPERIOR SERVER-SIDE AD INSERTION SOLUTION

thePlatform®

INTRODUCTION

The number of videos watched online is exponentially increasing and media companies are having more success than ever before at monetizing this growth with video advertising.[1] However, the proliferation of ad blockers is also on the rise. Ad blocker usage grew by nearly 70% between June 2013 and June 2014.[2] Server-side ad insertion with mpx ensures advertising playback for any screen by enabling advertising playback even when ad-blocking technology is present. Our deeply integrated solution provides all the flexibility our customers require to realize their unique Video-on-Demand advertising strategies.

ADVERTISING SUPPORT IN MPX

mpx supports the two main ways to approach targeted, dynamic online advertising:

1. Client-Side – Ads are resolved through calls to an ad server from the player the viewer is using, and then stitched into the playback experience on the player.
2. Server-Side – The stream coming down to the player already has the dynamic ads inserted.

Although client-side advertising has been a very effective way for our mpx customers to monetize through the years, we've started to see them run into challenges with this approach (ad blockers and heavy plugin development for new devices are two big ones). As server-side advertising resolves these challenges, we're now encouraging our customers to use it as an alternative approach to online advertising. In this white paper you will learn how both approaches to online advertising work in mpx, how the server-side approach resolves some of the challenges of the client-side approach, and how our mpx Server-Side Ad Insertion solution broadly raises the industry's bar on server-side advertising.

HOW CLIENT-SIDE AD INSERTION WORKS

Assuming a customer has an ad server that's returning ad information in the VAST format, here's how client-side ad insertion works in mpx:

1. A viewer initiates a 'play' in the mpx Player
2. The mpx Player calls mpx to get the content that the viewer wants to play
3. mpx determines both the content and the ad server URL(s), and sends them to the mpx Player in a SMIL metafile
4. The mpx Player then parses the SMIL file and discovers the client-side ad URLs
5. Once the mpx Player finds the ad URLs, it calls the ad server
6. After receiving a call from the mpx Player, the ad server returns a VAST payload
7. Once the VAST payload is received, the mpx Player then parses through it and starts playing the ad
8. As the ad plays in the mpx Player, any impression or beacon URLs fire from the mpx Player for tracking purposes

---

1 *U.S. Digital Video Benchmark: Adobe Digital Index Q2 2014*
2 *"Adblocking Goes Mainstream" by PageFair and Adobe, 2014*

CHALLENGES WITH CLIENT-SIDE AD INSERTION

As previously mentioned, although client-side ad insertion is incredibly popular and a proven money-maker, it creates some challenges for our customers. Two major limitations to this online advertising approach are that it's susceptible to ad blockers and requires heavy plugin development for new devices.

Ad Blockers

Ad blockers are essentially browser plugins that have a configured list of regular expressions that match known ad server URLs. When a browser makes an ad request, if the request matches one of the ad server URLs, the ad blocker will apply a change to the page. Then the page will simulate an error from the ad server URL and/or attempt to remove an element from the page, thereby effectively blocking the ad. According to a recent report by PageFair and Adobe, the number of viewers turning on ad blockers is increasing. Last year, approximately 28% of U.S. browser-users had an ad blocker turned on.[3] Why is this such a big limitation to client-side ad insertion? When an ad blocker is turned on, the mpx Player's call to the ad server will fail and no ads will be played. Although the mpx Player can detect this failure, prevent playback of the main video, and tell the viewer why playback was blocked, few of our customers have chosen to do this. Most customers would rather have their viewers engaging and sharing content, even if those ad-blocked views aren't being monetized. That said, missing the opportunity to monetize 28% of viewers is a tough pill to swallow.

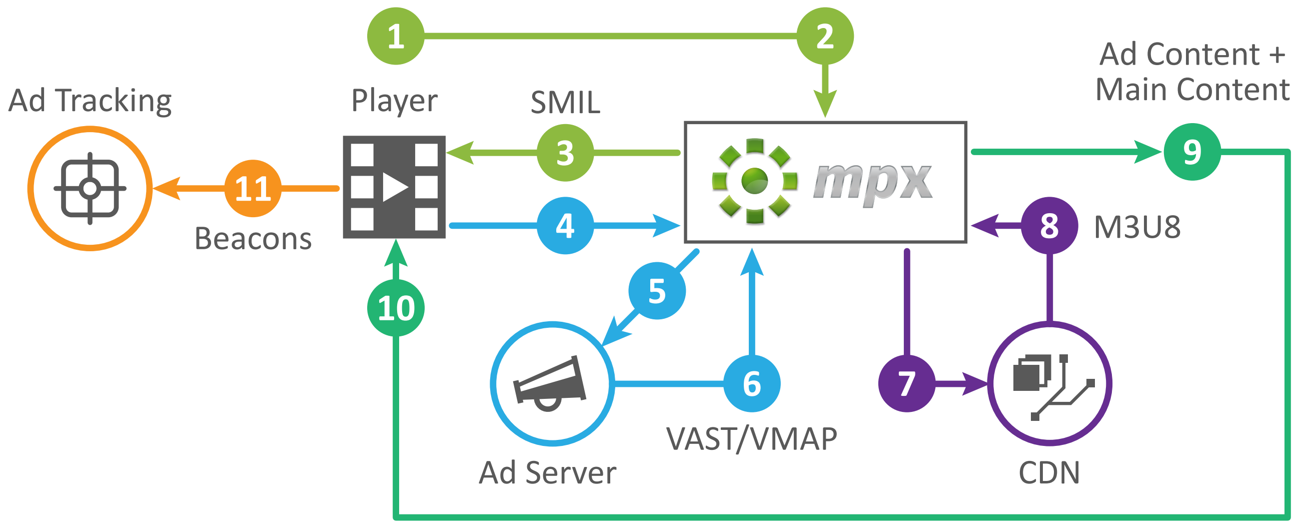
Heavy Plugins

In order for client-side advertising to work on new platforms, customers need to build new plugins for those platforms. These plugins tend to be "heavy," because they need to do all the negotiation with the ad server and manipulate the client stream. They also take a good amount of time to build. Most plugin development teams have limited resources, so using client-side advertising limits how quickly media companies can expand to new platforms. This ultimately impacts ad reach. As viewers gravitate to the latest and greatest devices, and away from the dinosaurs of yesterday, media companies slow to embrace new platforms will rapidly see a reduction in the views they can monetize.

3 *"Adblocking Goes Mainstream" by PageFair and Adobe, 2014*

HOW SERVER-SIDE AD INSERTION WORKS

Here's how server-side ad insertion works in mpx:



1. A viewer initiates a 'play' in the mpx Player
2. The mpx Player makes a request to mpx for the content URL. If server-side ad insertion is supported, instead of returning back the link to the video and to ad server URLs, mpx will return back a link to our server-side ad insertion service. We call this the "manifest service."
3. mpx will return this link in a SMIL file, which contains the manifest link, as well as metadata about the content, and a link to the "sidecar" file containing metadata about ad timing and placement.
4. The mpx Player will then call both the manifest link and the sidecar link. Both links are signed and cannot be tampered with or called from a different IP address than the request during step two outlined above. mpx will validate the request and then get the content and ad server URLs associated with the request. It will also get a non-identifying cookie that represents the viewer/user.
5. mpx will get the VMAP and/or VAST URLs on the ad policy, and call the ad servers to resolve them. If the original request in step two had a "debug=true" flag, the actual ad server URLs and responses will be echoed back in the sidecar file. These requests are made with an "X-Forwarded-For" header of the user's IP address, as well as the user's user-agent. The ID representing the user can also be passed through.
6. The ad server will return VMAP and/or VAST payloads, and mpx will parse them. If the ad payloads point to other ad servers, mpx will call them as well. Just like with client-side ad insertion, fail-over ad server URLs will be called if the initial ad server calls returned errors or empty payloads. If an ad isn't available as HLS, but is available as MPEG4 or another encodable format, mpx will ingest that ad at the highest available quality and submit an ingest and publishing request into mpx's workflow queue. When the file is available as HLS, the next request that returns that ad will use the converted file instead.

7. mpx now calls the CDNs that store each HLS link, for both the content and any individual ads, to fetch the M3U8 metafiles. It does NOT fetch the individual MPEG2 chunks: those continue to be delivered from the CDN in all cases.
8. The M3U8 files are returned from the CDN and parsed. If the request in step two specified a bitrate range or a maximum height and width for playback, the available bitrates will be pared back to just the matching set. At this point, mpx will combine the content HLS and ad HLS, dealing with encrypted content vs. non-encrypted ads, as well as finding the best match when the content is at different bitrates than the ads.
9. mpx returns back both a unified M3U8 that contains both content and ads as individual transport stream chunks as well as a sidecar file with metadata about ad positioning and other client-side configuration, like companion banners and tracking beacons.
10. The mpx Player parses the sidecar file and starts playback on the main content. Thanks to the sidecar file, the mpx Player is able to show indications for where midrolls appear in the timeline as well as ad count-downs while ads play. The mpx Player can also prevent skipping ads, if that was the directive from the ad server.
11. The sidecar also contains beacons and impression URLs, and the mpx Player will fire them off when they appear on the timeline. If the customer has an ad blocker turned on, these impressions can get lost, but the video will still play.

RESOLVING CLIENT-SIDE CHALLENGES

Server-side ad insertion mitigates the two main challenges of client-side ad insertion, ad blockers and heavy plugins:

Avoiding Ad Blockers

If there's no generic pattern to the ad transport stream URLs, as outlined in the previous section, ad blockers will not be able to block them. Although ad blockers could block the main manifest request, they wouldn't because they would also block basic video playback. Manifest formats typically have some metadata that say when the source for a manifest is changing, e.g., the "EXT-X-DISCONTINUITY" tag in the HLS format above. But again, ad blockers would not trigger off of these tags because they are sometimes used for normal content transitions in a playlist. So… even if a viewer has an ad blocker turned on, they'll still get shown video ads via server-side ad insertion.

That all said, the parts of an ad experience that must be inserted on the client-side, such as companion banners or overlays, can still be blocked. And, it's possible that an ad blocker can prevent impressions and other tracking from being fired off the mpx Player. Still, video ads won't be blocked and views will be monetized. It's also important to note that ad blockers continue to invest in new ways to block ads, so there's a bit of an arms race going on. For now, using server-side ad insertion effectively leap-frogs the current generation of ad blockers.

A Simpler Device Story

If a viewer has a device that natively understands the manifest format used, a media company doesn't have to do any extra plugin work for video ads to play. Just send the manifest URL to the device, and the mpx Player plays it (a compilation of content and ads). No plugin required.

For a richer experience, where fast-forwarding ads is prevented and both companions and client-side impressions are supported, there is still some client-side ad insertion work required. However, it's much less work than trying to build an entire client-side video ad insertion plugin since it doesn't touch playback or try to manipulate the video stream. The little work required is the creation of a metadata sidecar file in tandem with a light-weight plugin to parse it and send messages to different components to handle additional collateral and beacons. Since the implementation of a richer server-side ad insertion experience via the client-side is a fraction of the work of building a complete client-side ad insertion plugin, media companies can expand to new platforms much more quickly and economically.

RAISING THE SERVER-SIDE BAR

thePlatform's primary goal with the mpx Server-Side Ad Insertion solution was to solve the ad blocker and heavy plugin issues that arise when our customers use client-side ad insertion. Addressing those two challenges is the minimum bar for any server-side ad insertion technology today. However, here at thePlatform, meeting the minimum isn't good enough. For over a decade, we have strived to raise the bar on online video publishing and management. When building the mpx Server-Side Ad Insertion solution, we set out to exceed the industry's standard solution by incorporating six extra benefits:

1. We didn't want to require customers to condition content through our systems. We provide great tools for creating manifest formats from sources, but we also wanted to allow any manifest source as a content source, even CDNs like Akamai HDN that do just-in-time packaging of source files to create manifests.
2. If an ad server returns an ad in a non-manifest format, we wanted to automatically convert it to a format that can be inserted in a manifest. This is because many ad networks include redirects to third-party ad systems where they can't control the available formats. In addition, some of our customers might have many properties returning incompatible ads and therefore wanted to let multiple accounts share a common account for converting ads. Lastly, some other solutions in the marketplace handle this issue via a just-in-time packaging approach where the ad URL follows a pattern that can be detected by an ad blocker. We wanted our server-side solution to avoid that pitfall.
3. We wanted to allow any ad server that returns VMAP or VAST ad metadata to be used for server-side insertion. This would give our customers more options for ad partners, instead of forcing a particular integration.
4. We wanted to have a single delivery URL that could work for either client-side or server-side ad insertion, and could switch between the two modes depending on whether a manifest format is supported on the client or not.

5. We wanted to give customer ad ops teams a debug mode that gives them visibility into what calls are being made on the server and what's being returned.
6. We wanted to build on the pre-existing mpx infrastructure, so that server-side ad insertion was truly integrated with our core systems. We wanted customer implementation to be as simple as flipping a switch to "on" in their existing ad policy configuration.

thePlatform Can Help

thePlatform has always been a customer-driven company in that we look at their challenges and make it a priority to help them overcome those challenges. Server-side ad insertion solves two of our customers' biggest challenges with client-side advertising – ad blockers and heavy plugin development. From a broader perspective, we built our mpx video publishing and management system to be flexible so that it can always accommodate the next generation(s) of TV-enablement services. Research clearly shows that more and more TV viewers are consuming their premium content online. Right along with viewers, all the rules and policies that govern TV are also moving online… starting with un-blockable advertising and then progressing into geographic blackouts, emergency action messages, personalized channels, automatic fail-over to back-up streams, alternate streams, and other dynamic content replacement scenarios. We expect the foundational work we've done with the mpx Server-Side Ad Insertion solution to extend to all of these scenarios in the coming months and years. We look forward to helping your online TV business overcome the challenges of today, and those that will inevitably surface in the future.

Please feel free to give us a call to learn more about the mpx Server-Side Ad Insertion solution and how we can help your online TV business monetize:

Toll Free: 1 (877) 436-7940

RESOURCES

thePlatform's Technical Resource Center:

https://help.theplatform.com/

thePlatform's white papers and case studies:

http://www.theplatform.com/solutions

Have a comment on this paper?

whitepapers@theplatform.com

1000 Second Avenue, Suite 1000
Seattle, WA 98104
theplatform.com
North America: sales@theplatform.com
Asia/Pacific: APACsales@theplatform.com
EMEA: EMEAsales@theplatform.com