HOMAD
Highly **O**ptimized **M**achine **F**or **A**dvertisment **D**elivery

HOMAD Guide
2018-10-17

HOMAD Implementation Step-by-Step

HOMAD is a service for secure and reliable data communication between browsers and servers. The HOMAD technology modifies the path your network packages take when a direct communication between your users and your servers is not possible. This fixes problems like broken network routes, misconfigured firewalls, network blocking browser plugins (adblockers) and it solves problems of browser modes like tracking protection or privacy enhanced modes. This document describes how you can enable the HOMAD service for your video player and the corresponding ad delivery setup. The result should be a huge gain in ad fill rates while respecting your users privacy settings.

For the implementation of the HOMAD technology the following steps are mandatory and explained in this document:

SCHNEE VON MORGEN

0. Let's Talk

In most cases and all standard setups, HOMAD will work out of the box just by running the code on the corresponding webpage. But as the HOMAD scripts are on the critical path of content delivery, we would like to have an overview of all different player usages on your webpage. E.g. have you got a proprietary system in place to signal the videoplayer that no advertisement is to be played? Are you using any other scripts aside the player itself for content playback (DRM, AdTag building scripts, header bidding)?
HOMAD takes over the control of your video player in case it detects that an ad delivery through the content player has failed. To be able to do that, HOMAD will need access to the following functions and information: play/pause, video duration, playhead position or current time, autoplay on/off. Furthermore HOMAD needs to be able to add event listeners to all player and - more important - ad framework events. But in most cases none of that will be of your concern, because the HOMAD service covers a lot of standard and non-standard use cases for video players.

1. Setup DNS Configuration

These DNS entries (wildcard CNAMEs) need to be configured: (this needs to be done per Client URL)

```
*.1.damoh.[EXAMPLE.COM] --> 1.examplecom.damoh.schneevonmorgen.com
*.2.damoh.[EXAMPLE.COM] --> 2.examplecom.damoh.schneevonmorgen.com
*.3.damoh.[EXAMPLE.COM] --> 3.examplecom.damoh.schneevonmorgen.com
```

For example.tv this results to:

```
*.1.damoh.example.tv --> 1.exampletv.damoh.schneevonmorgen.com
*.2.damoh.example.tv --> 2.exampletv.damoh.schneevonmorgen.com
*.3.damoh.example.tv --> 3.exampletv.damoh.schneevonmorgen.com
```

If you are running a multi-language and multi-continent website, please geo-target your DNS settings like this:

```
*.1.damoh.de.example.tv --> 1.deexampletv.damoh.schneevonmorgen.com
*.2.damoh.de.example.tv --> 2.deexampletv.damoh.schneevonmorgen.com
*.3.damoh.de.example.tv --> 3.deexampletv.damoh.schneevonmorgen.com
*.1.damoh.us.example.tv --> 1.usexampletv.damoh.schneevonmorgen.com
*.2.damoh.us.example.tv --> 2.usexampletv.damoh.schneevonmorgen.com
*.3.damoh.us.example.tv --> 3.usexampletv.damoh.schneevonmorgen.com
```

In the above examples, we are configuring three DNS entries that point to three server clusters on our end. One cluster can handle approx. 500.000 video views per day and three clusters are the minimum configuration for every customer.

If your website has significantly more traffic than this, you can calculate the number of needed clusters like this:

HighTrafficWebsite.com has 5 Million video views a day in the configured region
5.000.000 / 500.000 = 10
please configure your DNS for 10 server clusters

```
*.1.damoh.hightrafficwebsite.com --> 1.hightrafficwebsitecom.damoh.schneevonmorgen.com
*.2.damoh.hightrafficwebsite.com --> 2.hightrafficwebsitecom.damoh.schneevonmorgen.com
*.3.damoh.hightrafficwebsite.com --> 3.hightrafficwebsitecom.damoh.schneevonmorgen.com
*.4.damoh.hightrafficwebsite.com --> 4.hightrafficwebsitecom.damoh.schneevonmorgen.com
*.5.damoh.hightrafficwebsite.com --> 5.hightrafficwebsitecom.damoh.schneevonmorgen.com
*.6.damoh.hightrafficwebsite.com --> 6.hightrafficwebsitecom.damoh.schneevonmorgen.com
*.7.damoh.hightrafficwebsite.com --> 7.hightrafficwebsitecom.damoh.schneevonmorgen.com
*.8.damoh.hightrafficwebsite.com --> 8.hightrafficwebsitecom.damoh.schneevonmorgen.com
*.9.damoh.hightrafficwebsite.com --> 9.hightrafficwebsitecom.damoh.schneevonmorgen.com
*.10.damoh.hightrafficwebsite.com -> 10.hightrafficwebsitecom.damoh.schneevonmorgen.com
```

If you are running a high-load-multi-continent website let’s have a talk on the best and most simple DNS setup. We can also deploy NS delegation setups under those complex scenarios.

2. Setup the Configuration Files

Every client will be able to activate/deactivate HOMAD via a JSON configuration file. In addition this file configures the list of HOMAD servers the plugin will communicate to as well as the internal aliases.

We’ve already configured a file for your use case and it should be attached to this document:

`homadConfig[YOUR URL].json`

This file will only work, when the DNS changes (see above) are done.

For testing purposes you can use this file:

`homadConfig[YOUR URL]-TEST.json`

This file calls schnee von morgen servers and will not work in a productive environment because it is intended for testing purposes only. This file should be hosted publicly readable on the client domain. Also be aware to set the CORS headers accordingly.

We also created a **HOMAD alias** for the ad server:

`[https://YOUR.ADSERVER.COM/]`

which is named:

`ALIAS: "[tvexampleyouradservercom]"`

and included it in the client configuration file. If you want to test against other adservers we will have to change this alias configuration.

3. Implementing HOMAD

The HOMAD HTML5 implementation does not require or relies on any 3rd party javascript libraries. It will run smoothly within all modern browsers that support the use of media resources such as audio or video through their corresponding tags <audio> and <video>.

The HOMAD HTML5 implementation consists of two parts: **homad.js** and **hd-main.js**.

homad.js

The homad.js file will be hosted by you under your domain. It contains the HOMAD configuration and the generic adTags that will be used by our implementation:

```
...
var conf = {
    // add the location of your config file
    clientConfig: 'https://[YOUR-CLIENT-CONFIG]',
    // add the ad tag you are using (string or function returning the adtag)
    adTag: 'https://[YOUR-ADTAG]',
    ...
};
...
```

You have to change the configuration part (the conf object) of the script according to your needs. The clientConfig and adTag entries in the original file point to a test configuration and a test ad tag that have to be changed to your config file and your ad tag.

The script is enclosed in an anonymous function and is executed immediately. It will check the configurations on the customers side and the availability of the SVM services. On overall success it will fetch the hd-main.js and pass the configuration through a custom event to the homad implementation.

It is essential that the homad.js script is loaded and executed as soon as possible, meaning a "long time" before the content player initialization.

To prevent blocking of this initial script you have to append the homad.js script to another script file, that is essential for the site or include it inline into your HTML document. If you intend to inline the homad.js please notice that it should be inlined into an existing script tag that is used on your side and is essential for that site functionality. That way inline script tag filtering (e.g. https://github.com/gorhill/uBlock/wiki/Inline-script-tag-filtering) can be prevented.

You can download the homad.js script here:

https://hgc-cf-cache-1.svonm.com/homad.js

hd-main.js

The hd-main.js script will be automatically included by homad.js. It will run several checks to identify the usage of an adBlocker (element hiding, adtag tests, and 3rd party cookies). It will further detect the video player that is used by the site.

If an adBlocker is detected and the display of the video ad has failed, hd-main.js will pause the content player and initialize a homad display directly on top of the original player (yes, we support adjusting to the content player size, on the fly resizing and fullscreen).

The ad configured through the pre/postrollAdTag configuration will be delivered in the homad display, which is then destroyed and the original content playback is resumed.

Configuration options

configuration files

* **clientConfig**: location of your config file, e.g. https://example.com/config.json. It can either be a url pointing to your config file, or a function that returns your configuration object. The function will be called repeatedly until a configuration object is available or until a maximum time span of 5 seconds.
* **globalConfig**: global config url, this will be set by us
* **noClientConfigCacheBuster**: if this flag is true, HOMAD will not use any cache buster technics when loading the client configuration file, which is hosted on your side.

callbacks

* **onReady**: callback that will be called as soon as HOMAD is ready to go.
* **onFailure**: penalty function that will be called if you do not want to use the built-in penalty function. But be aware that content playback MUST be impossible for the user when the penalty is called.

ad tags

* **adTag**: ad tag url, which **will be used for pre- and postrolls** if nothing else is specified, e.g. https://example.com/vast.xml. Setting this to false will skip the ad delivery. If the adTag is not available at script start, you can use a function that returns an adTag. This function will be called after HOMAD takes over the ad delivery, meaning after the player and/or the adFramework failed to deliver an ad. It will be provided with the player instance as HOMAD sees it as the first parameter. @see adTag-Filter examples
* **prerollAdTag**: use this if you want to use different ad tags for pre/post rolls, e.g. https://example.com/prerollvast.xml. Setting this value to false will skip the pre roll, if adTag is set to false, too.
* **postrollAdTag**: use this if you want to use different ad tags for pre/post rolls, e.g. https://example.com/postrollvast.xml. Setting this value to false will skip the post roll, if adTag is set to false, too.
* **midrollAdTag**: use this if you want to use different ad tags for midrolls, e.g. https://example.com/midrollvast.xml. Setting this value to false will skip the mid roll, if adTag is set to false, too.
* **yieldlab:** use this to configure headerbidding through yieldlab, make sure you have a [yieldlab]-macro configured in your adTag. @see yieldlab example

behavior

* **maxWrapperDepth**: this is the maximum wrapper depth HOMAD will follow. After exceeding this limit is reached HomadEvent.wrapperLimitReached will be fired and the content video will continue to play.
* **skipable**: if set to true, skipoffsets will be respected
* **adjustAdVolumeToContentPlayer**: if set to true, homad display inherits volume from content video player
* **sideloads**: any additional request that needs to be fired. Takes an array of maps, each map must be given two keys 'alias' and 'url' (ex. 'sideloads': [{'alias': 'mydomainalias', 'url': ‘https://mydomain.com/api'}]).
* **skipAdOnContextChange**: In case of a single page app, that is reusing the content player for every new video (e.g. jw player playlist setups) setting this property to true will skip the currently playing ad when changing the content player video source.
* **viewableImpression**: use this property if you want to change the viewability definitions. The default value is: viewableImpression: {'percent' : 50, ,'timeMS' : 2000}

styling

* **pauseButton**: innerHTML of div, use this config param to style your button.
* **playButton**: innerHTML of div, use this config param to style your button.
* **muteButton**: innerHTML of div, use this config param to style your button.
* **unmuteButton**: innerHTML of div, use this config param to style your button.
* **admessage**: This is a message that will be displayed ontop of the ad display. Use "[time]" as a placeholder for the remaining time in seconds of the ad video. The standard text value is: "Werbung endet in [time] Sekunde(n)". This property can be configured as a function which has to return the expected string, too. The function will be called with the detected player as first parameter.
* **zIndex**: zIndex of the HOMAD AdPlayer. You can set this in case you experience an overlay issue.
* **positionIfFixed**: HOMAD will render it's display ontop of the content player. If the content player is fixed (or placed in a fixed environment), you might want to specify a position attribute value if the displays are not aligned.
* **runInsideContainer**: if set to true, HOMAD will inject its ad display into the detected player container which will turn the HOMAD display to an adjacent sibling of the content video element.
* **video.style.position**: if set, the HOMAD display will use this position style attribute for its video tag
* **video.style**: if set, the HOMAD display will use this value for its video tag style attribute. It will override video.style.position option, if set.
* **video.attributes**: add attributes onto the HOMAD video tag. {"playsinline":"","poster": "https://example.com/poster.png"}
* **useBoundingClientRect**: switch between different positioning methods, if set to true, [element].getBoundingClientRect will be used.
* **skipableButton**: innerHTML of div, use this to style your button, [time] will be replaced with the remaining time until skip is possible
* **skipButton**: innerHTML of div, use this to style your button
* **innerWrapper.style**: if set, the style property of the innerWrapper div will be set to this value
* **caption.style**: if set, the style property of the caption div will be set to this value
* **controls.style**: if set, the style property of the controls div will be set to this value
* **playPause.style**: if set, the style property of the playPause div will be set to this value
* **mute.style**: if set, the style property of the mute div will be set to this value

* **skip.style**: if set, the style property of the skip div will be set to this value

outstream

* **inread.elementID**: Specifies the element which the inRead player is observing.
* **inread.width**: Width of the inread player in px.
* **inread.height**: Height of the inread player in px.
* **inread.minVis**: Minimal area that has to be visible of the inread player, range from 0-100.
* **inread.positioning**: before, after, replace in respect to inread.elementID

pureHOMAD

HOMAD can be used as the sites main player. You can activate this by using the player property. This property is an array and can contain multiple player definitions. The player is controlled by triggering defined events on a container element.

* **player[index].containerID**: Specifies the element which will be used to place the HOMAD display.
* **player[index].autostart**: boolean value, true indicating autostart
* **player[index].playEvent**: name of the playEvent used to start playback, has to be triggered on the element with containerID as the id attribute.
* **player[index].pauseEvent**: name of the pauseEvent used to pause playback, has to be triggered on the element with containerID as the id attribute.
* **player[index].resumeEvent**: name of the resumeEvent used to resume playback after pause, has to be triggered on the element with containerID as the id attribute.
* **player[index].muteEvent**: name of the muteEvent used to mute playback, has to be triggered on the element with containerID as the id attribute.
* **player[index].unmuteEvent**: name of the unmuteEvent used to unmute playback, has to be triggered on the element with containerID as the id attribute.
* **player[index].***: as the player configuration is passed to the adTag function, you can add any property you need.

Examples

yieldlab (header bidding) example:

```
...
var conf = {
    // add the location of your config file
    clientConfig: 'https://[YOUR-CLIENT-CONFIG]',
    // add the ad tag you are using (string or function returning the adtag)
    adTag: 'https://[YOUR-ADTAG]',
    ...
};
...
```

This configuration will make a call to yieldlab configured by the url property. We will provide you with the alias, which has to match with our backend configs. You can configure as many Yieldlab calls as you like. HOMAD will enhance the placeholder property with the corresponding Yieldlab answers. As for now HOMAD supports: [id], [price], [advertiser], [curl], [format], [src], [pricerange], [pid], [prio], [did]. The macro will then be replaced with the enhanced placeholder string. If there is no valid Yieldlab response (eg. empty answer or any error) the macro will be replace with an empty string.

adTag-Filter example:

ENCODE: this filter can be used to apply encodeURIComponent to the string surrounded by the brackets. This will also work with nested calls and can be used inside the yieldlab configuration placeholder property.

```
adTag: 'https://example.com/vast.xml?foo=bar&bar=ENCODE(foo=true)'

=>

adTag: 'https://example.com/vast.xml?foo=bar&bar=foo%3Dtrue'
```

HOMAD Events

HOMAD will fire custom events on the window object with the name "hdEvent" that can be listened to by e.g.:

```
window.addEventListener('hdEvent', function(evt){
    console.log(evt);
});
```

The event details can be found in the event's details property. The following events are available:

```
HomadEvent.penaltyEvent = {
    // penalty function has been called, due to blocked functionality
    'code': '1000',
    'type': 'penalty',
    'message': '',
    'name': 'penalty'
};
```

```
HomadEvent.noMediaEvent = {
    // the media files could not been loaded, e.g. network error, no files specified
    'code': '1001',
    'type': 'mediaerror',
    'message': '',
    'name': 'adError'
};
```

```
HomadEvent.vpaidEvent = {
    // HOMAD recieved a VPAID response
    'code': '1002',
    'type': 'vpaid',
    'message': '',
    'name': 'vpaid'
};
```

```
HomadEvent.emptyVastEvent = {
    // HOMAD recieved an empty VAST
    'code': '1003',
    'type': 'emptyVast',
    'message': '',
    'name': 'adError'
};
```

```
HomadEvent.xmlMalformattedEvent = {
    // HOMAD recieved a malformatted VAST response
    'code': '1004',
    'type': 'malformattedXML',
    'message': '',
    'name': 'adError'
};
```

```
HomadEvent.adStartEvent = {
    // HOMAD ad delivery started
    'code': '1005',
    'type': 'adStart',
    'message': '',
    'name': 'adStart'
};
```

```
HomadEvent.adFirstQuartileEvent = {
    // HOMAD ad delivery first quartile
    'code': '1006',
    'type': 'adFirstQuartile',
    'message': '',
    'name': 'adFirstQuartile'
};
```

```
HomadEvent.adMidPointEvent = {
    // HOMAD ad delivery mid point reached
    'code': '1007',
    'type': 'adMidPoint',
    'message': '',
    'name': 'adMidPoint'
};
```

```
HomadEvent.adThirdQuartileEvent = {
    // HOMAD ad delivery third quartile
    'code': '1008',
    'type': 'adThirdQuartile',
    'message': '',
    'name': 'adThirdQuartile'
};
```

```
HomadEvent.adCompleteEvent = {
    // HOMAD ad delivery completed
    'code': '1009',
    'type': 'adComplete',
    'message': '',
    'name': 'adComplete'
};
```

```
HomadEvent.adImpressionsCalledEvent = {
    // HOMAD ad delivery impressions called
    'code': '1010',
    'type': 'adImpressionsCalled',
    'message': '',
    'name': 'adImpression'
};
```

```
HomadEvent.vastLoadingFailedEvent = {
    // HOMAD could not successfully load vast
    'code': '1011',
    'type': 'vastLoadingFailed',
    'message': '',
    'name': 'adError'
};
```

```
HomadEvent.noCreativeEvent = {
    // VAST document does not contain any creative/media files
    'code': '1012',
    'type': 'noCreative',
    'message': '',
    'name': 'adError'
};
```

```
HomadEvent.emptyVastFromHomadServerEvent = {
    // HOMAD recieved an empty VAST from HOMAD Server
    'code': '1013',
    'type': 'emptyVastFromHomadServerEvent',
    'message': '',
    'name': 'adError'
};
```

```
HomadEvent.wrapperLimitReached = {
    // HOMAD recieved an empty VAST from HOMAD Server
    'code': '1014',
    'type': 'wrapperLimitReached',
    'message': '',
    'name': 'adError'
};
```

```
HomadEvent.clickthroughEvent = {
    // HOMAD detected click on ad viedeo
    'code': '1015',
    'type': 'clickthrough',
    'message': '',
    'name': 'adClick'
};
```

```
HomadEvent.continueContentEvent = {
    // resume content video
    'code': '1016',
    'type': 'continueContent',
    'message': '',
    'name': 'continueContent'
};
```

```
HomadEvent.contentPlayerPlay = {
    // HOMAD called play on content player
    'code': '1017',
    'type': 'contentPlayerPlay',
    'message': '',
    'name': 'contentPlayerPlay'
};
```

```
HomadEvent.contentPlayerPause = {
    // HOMAD called pause on content player
    'code': '1018',
    'type': 'contentPlayerPause',
    'message': '',
    'name': 'contentPlayerPause'
};
```

```
HomadEvent.adSkipped = {
    // ad has been skipped
    'code': '1019',
    'type': 'adSkipped',
    'message': '',
    'name': 'adSkipped'
};
```

```
HomadEvent.adRequest = {
    // ad is requested by the player
    'code': '1020',
    'type': 'adRequest',
    'message': '',
    'name': 'adRequest'
};
```

```
HomadEvent.adPlay = {
    // ad is played
    'code': '1021',
    'type': 'adPlay',
    'message': '',
    'name': 'adPlay'
};
```

```
HomadEvent.adPause = {
    // ad is paused
    'code': '1022',
    'type': 'adPause',
    'message': '',
    'name': 'adPause'
};
```

```
HomadEvent.adTime = {
    // ad playback is in progress
    'code': '1023',
    'type': 'adTime',
    'message': '',
    'name': 'adTime'
};
```

```
HomadEvent.adMeta = {
    // adID and real Impression URL
    'code': '1024',
    'type': 'adMeta',
    'message': '',
    'name': 'adMeta'
};
```

```
HomadEvent.viewable = {
    // viewable impression
    'code': '1025',
    'type': 'viewable',
    'message': '',
    'name': 'viewable'
};
```

```
HomadEvent.notViewable = {
    // not viewable impression
    'code': '1026',
    'type': 'notViewable',
    'message': '',
    'name': 'notViewable'
};
```

```
HomadEvent.mute = {
    // ad muted
    'code': '1027',
    'type': 'mute',
    'message': '',
    'name': 'mute'
};
```

```
HomadEvent.unmute = {
    // ad unmuted
    'code': '1028',
    'type': 'unmute',
    'message': '',
    'name': 'unmute'
};
```

```
HomadEvent.contentPlayerMuted = {
    // content player was muted at ad start
    'code': '1029',
    'type': 'contentPlayerMuted',
    'message': '',
    'name': 'contentPlayerMuted'
};
```

Every event will be populated with a state attribute indicating the state (setup/preroll/postroll) HOMAD has been executing at that time. In Addition, the content player's video and container tag will be added, too. Also the events will be enriched with information about the currently playing ad, if those information is available, eg. duration, currentTime and adParameters.

4. HOMAD over HTTPS

All HOMAD configuration resources have valid SSL certificates. We are going to generate and serve Let's Encrypt certificates for your configured DNS entries pointing to our clusters and we will also renew those certificates when needed.

Adapting the configuration for SSL

1. If you have been using HOMAD prior to Oct. 2016 make sure you have the latest version of homad.js.
2. Make sure that all URLs in your clientconfig (the configuration file that is hosted by you) are using HTTPS.
3. Make sure that all URLs in the conf-part of homad.js (especially the adTag) are using HTTPS.

5. Working with HOMAD - Testing, Debugging and Launch

In order to test your HOMAD implementation simply install an ad blocking plugin of your choice, make sure you are using the latest filter list, activate it and load your page. Now you should see your ads even with the plugin being activated. To get a closer look or when you have to debug problems please survey the network connections using the developer tools provided by your browser. By looking at the response header **X-Homad-Url**, you can identify the URL HOMAD is calling for you.

When moving into production we are able to active HOMAD only for a certain percentage of your users providing you a A/B testing opportunity. Please contact us for further information about it.

6. Contact

For further questions concerning the HOMAD implementation don’t hesitate to contact Vincent or Johannes:

Vincent Schreiter
schreiter@schneevonmorgen.com
+49 173 53 44 391

Johannes Hofmann
hofmann@schneevonmorgen.com
Skype: hqawnenretsz