SAS® 360 Match

Request Application Programming Interface

Updated October 30, 2023

Contents

Request Application Programming Interface

SAS® 360 Match is a flexible advertising delivery platform able to serve online display, video, and mobile advertising channels. This Application Programming Interface (API) contains directives, protocols, and methods for working with and requesting ads with SAS® 360 Match.

Web Server Ad Call Directives

As a special-purpose web server, the SAS® 360 Match ad server responds to HTTP requests from any client. It uses path information to determine which creative to serve. The method in this path contains the directive that tells the server how to return the creative, as in the following example:

```
http://shortname-ads.aimatch.com/{shortname}/hserver/site=A/size=728x90
```

There are multiple server directives that you can use to develop ad calls that are appropriate for your server model. In this case, the directive `hserver` determines the HTTP content type that the server returns. In the case of `hserver`, it is text/html. The `jserver` directive returns application/x-javascript.

Ad Request Responses

HTTP requests sent to the request API receive an HTTP 200 OK response, unless otherwise noted. The following HTTP headers are in the response:

Header	**Value**
Cache-Control	no-cache, no-store, max-age=0, must-revalidate
Connection	keep-alive
Content-Encoding	gzip
Content-Length	<varies>
Content-Security-Policy	default-src 'self'
Content-Type	<varies>
Date	<varies>
Expires	-1
Pragma	no-cache
Server	<varies>
Set-Cookie	<varies>, `Secure`, and `HttpOnly` flags optional.
Strict-Transport-Security	max-age=31536000; includeSubDomains
X-Content-Type-Options	nosniff
X-XSS-Protection	1; mode=block

Note: The `Content-Type` header value is dependent on the ad request method and, when using the TSERVER directive, the MIME type specified for the creative format.

If the request is a CORS (cross-origin resource sharing) type, CORS headers are sent in response:

Header	**Value**
Access-Control-Allow-Credentials	true

Access-Control-Allow-Headers	X-Requested-With, origin, content-type, accept, accept-encoding, accept-language, cache-control, dnt
Access-Control-Allow-Methods	GET, OPTIONS
Access-Control-Allow-Origin	<varies>
Access-Control-Max-Age	600

Cookies

Each request sent to SAS® 360 Match is checked for a MID cookie. The MID cookie uniquely identifies each visitor and establishes a visitor session that many ad-serving features rely on. The MID cookie can be renamed.

The MID cookie supports `HttpOnly` and `Secure` flags. Contact SAS Technical Support at support@sas.com to configure the cookie, rename the cookie, or enable the flags.

Cookie exchange can be disabled. See the section on Disabling Cookies in the Advanced Features Guide at http://support.sas.com/documentation/prod-p/iap/default/en/PDF/Advanced_Features_SASIA.pdf.

LSERVER Redirect

If an ad request does not contain a MID cookie in the HTTP headers, the response is a 302 Redirect to the initial request URL, except when `lserver` is prepended to the directive (for example, `lserver/hserver`). The 302 Redirect response includes a Set-Cookie header with the MID cookie defined. If the visitor responds to the LSERVER request with the MID cookie, no future redirects are done. If the visitor does not respond with the MID cookie, the next ad request follows the same pattern by redirecting to LSERVER. In both situations, ads are returned as normal.

If any other type of request does not contain a MID cookie in the HTTP headers, the response contains a Set-Cookie header with the MID cookie, but no redirect is done.

Defaults

An engine default, or simply default, is a standard response provided by the server when it does not find a matching flight creative to serve after evaluating the ad queue. You can avoid serving an engine default ad by specifying your own default ad in the last tier as the creative to serve should nothing else match an ad request. In this case, the ad that is served is not recorded as a default but rather as the creative that you trafficked.

An engine default is also served when a network creative is selected but encounters a time-out or error when communicating with the third-party server that provides its response.

When an engine default is served, the engine normally sends the following response:

```
<a href="http://shortname-ads.aimatch.com/shortname/adclick/FCID=-4/<targeting
path info>" target="_top"><img src="http://content.aimatch.com/default.gif" alt=""
border="0" ></a>
```

Defaults are indicated by an FCID value of “-4” in the response.

However, you can override this default response by including /ADDEFAULT=<format>/ in the request or creating a creative format named `addefault`. Whatever is defined in this format is

used as the content for an engine default. If ADDEFAULT is specified in the request, it overrides the use of the creative format named `addefault`, if it exists.

A different type of default can serve when there is an internal error handling the entire request, such as an internal time-out when a server does not respond. Normally, the engine sends the following response:

```
<a href=http://www.aimatch.com/ target="_top"><img
src=http://content.aimatch.com/default.gif border="0"></a>
```

Similar to an engine default, you can override this default response by including /DEFAULT=<format>/ in the request or creating a creative format named `default`. Whatever is defined in this format is used as the content for this type of error.

The recommended practice is to use the same format for `default` and `addefault` for requests for a single ad, such as tserver, hserver, or jserver requests. For batch requests, such as bserver and dserver requests, use `default` for the failed request and `addefault` for specific ads that might result in engine defaults.

Contact SAS Technical Support to customize the URLs for the anchor HREF value and image source.

Error Responses

Here are some common error responses and their definitions:

- **404 Invalid Job ID** when the specified job ID does not exist.
- **408 Request Timeout** when the engine fails to respond in time to a request.
- **429 Too Many Requests** when the server receives too many requests from a visitor. The response is sent until the condition subsides.
- **500 Internal Server Error** when there is an uncategorized error.

Ad Tag Parameters

The following ad tag parameters enable you to change simple functionality without revising your creatives. Note that ad tag parameters are not case sensitive (for example, /AREA=4 is the same as /area=4).

The following ad call parameters are key-value pairs that the engine uses to select ads:

area

passes in the area value for targeting.

custom tag name

can be included in your ad calls for logging or targeting if you have created custom tags and associated values. Pass in the exact tag name and value. For example, if you have designated

a tag called "position" that has three values (top, middle and bottom), the ad tag might contain /position=top/.

duration

specifies the maximum time duration, in seconds, to be filled with multiple ads in a dserver ad request.

keyword

specifies one or more keywords used in targeting. Separate multiple keywords with a space or comma.

latitude and longitude

specifies the latitude and longitude, in degrees, for proximity targeting.

The latitude and longitude geographic lookup can update the location information for a visitor and replace geo tags such as GEO_CITY, GEO_COUNTRY, GEO_METRO_CODE, and GEO_REGION based on the latitude and longitude of the visitor. This feature is disabled by default. Contact SAS Technical Support to enable the latitude and longitude geo lookup feature.

The visitor's IP address establishes an initial value for geo tags when the session starts. Subsequent ad calls that specify latitude and longitude coordinates, /LATITUDE=X/LONGITUDE=Y, can change the geo tag values. The value for GEO_ZIP_CODE_TEXT does not change because that data is not associated with a geographic location.

When latitude and longitude geo lookup is enabled, an ad call that contains the latitude and longitude coordinates does the following:

- If the coordinate is less than the default distance of 1 km from the previous coordinate used for lookup, no lookup is done. Contact SAS Technical Support to change the default distance.
- If the distance from the previous lookup is large enough or if no previous lookup has occurred, the lookup happens.
- The lookup finds geographic data with the coordinate that is closest to the coordinate in the ad call.
- If the distance from the data point's coordinate is farther than the default distance of 50 km, the data is not used and the lookup has no effect. This likely indicates the data is insufficient in that area. Contact SAS Technical Support to change the default distance.
- If the distance is short enough, the GEO_CITY, GEO_COUNTRY, GEO_METRO_CODE, and GEO_REGION values for that data point override whatever values are held with the visitor already.
- These geo tags become the new values for the visitor for the session. They take the place of the previously known values until another coordinate is introduced. Subsequent

ad calls do not need to include coordinates if the geo tags do not change. Ad calls continue to use the geo tags produced by the previous coordinate lookup.

nocompanion

prevents selection of a companion flight during an ad call. The tag can occur anywhere in the URL after the ad call method directive, such as hserver or bserver, and must be surrounded by forward slashes (/). For example,

```
http://serve.adserver.com/client/hserver/site=X/area=1/size=300x250/NOCOMPANION/
```

Ad calls on a page should contain the /NOCOMPANION/ tag if a flight with a loose policy setting can deliver to the page and if the desired behavior is to prevent companion ads from delivering to any ad calls on the page. If the /NOCOMPANION/ tag is present in the first ad call on a page, a flight with a strict policy will not deliver to any ad calls on the page. The term page refers to a sequential set of ad calls with the same view ID.

pbfcid

used in passbacks.

random

prevents the caching of creatives, especially to third-party servers.

saspb

used in passbacks.

shortname

a unique string used to identify each SAS® 360 Match customer. The specific `shortname` value to be used for your implementation of the SAS® 360 Match ad tag is provided by the SAS Implementation Manager.

site

specifies an available site value for targeting. Ads can respond to a call that contains the specified site name if they have been targeted to that site or if there is no site value specified.

size

specifies an available ad size value for targeting. Separate multiple sizes in an ad call with a comma. For example, `size=300x250,160x600` means that ads with the size of either 300x250 or 160x600 can serve to the ad call.

viewid

notifies the system about which ad requests belong together.

Considerations for Developing an Ad Call

Choosing the best ad call for your situation depends on the targeting schema that you have created, the nature of your web pages, and the creative you are serving to the pages. Carefully consider these factors before choosing an ad call:

Caching and Cache Busting

Web browsers often save a local copy of files that they have requested and displayed to display them more quickly the next time it is requested. The saved version is used instead of re-requesting the file from its server. Some internet service providers (ISPs), network proxy servers, or other network components might use the same process. The process is known as caching and it is normally a positive enhancement to the web-browsing experience.

When an online advertisement is cached and then viewed, the ad server is not able to count the impression. To prevent this problem, each instance of a given ad call should contain a unique random number. The unique number causes the browser or proxy server to see it as a unique object and therefore make a fresh request for it from the ad server. The result is proper impression counts and revenue allocation. This process is known as cache busting, and there are several ways to effectively implement it. It is very important that cache-busting techniques be used properly.

Cache-Busting Techniques

SAS® 360 Match uses a specific ad call parameter designed to receive the random cache-busting number (`/random=`), although any unique variation in the ad-tag string can provide effective cache busting. The advantage of using the `/random=` parameter is that it can be referenced in several ways by objects that are downstream from the ad request.

The parameter can be used to provide cache busting within the source code of rich media or third-party advertisements without a separate, unique-number generation process. The random number can be generated by a server-side process and dynamically inserted into the ad call, or it can be generated using a client-side script that generates the entire ad call dynamically. Caching can lead to major discrepancies between SAS® 360 Match counts and those of third-party ad servers or site publishers.

The parameter prevents the caching of creatives, especially to third-party servers. You can prevent caching to maximize impression counts (rather than using a cached version of the ad from the browser or a proxy server). Any value passed in with the random ad call parameter is also used to populate the %%RANDOM%% token within ads that contain it.

Each ad call on a page should have a unique random value. If two or more ad calls on the page have the same targeting values and random values, the first ad delivered for rendering could be cached in the other ad call positions. In addition, the random value should be a different value for each page or page refresh. This is typically done dynamically using JavaScript. For example:

```
http://shortname-ads.aimatch.com/{shortname}/hserver/site=x/area=y/size=728x90
/pos=top/random=345678/
```

Companion Ads

The `/viewid=` tag is required if you are going to use companion ads (also called roadblock ads). The `viewid` parameter notifies the system which ad requests belong together. Using the same `viewid` value in a set of ad calls enables the system to treat the set of ad calls as a single page. The number specified as the `viewid` must be the same for each ad call in a group, which identifies a set of ad calls as existing on one page of a website. Also, the number must be unique to that service of the group (or page). This is typically accomplished dynamically, using JavaScript.

Specify `viewid` if you want to use tier settings to suppress duplicate flights or advertisers appearing on the same page. You can also use `viewid` if you want to suppress delivery of creatives from different advertisers appearing on the same page (according to their defined categories). For example:

```
http://shortname-ads.aimatch.com/{shortname}/hserver/site=x/area=y/viewid=
438943894343/size=728x90/pos=top/
```

Clash Management

You can communicate ad category clashes with a third party and gather data about categories the third party is providing in fulfilling ad breaks.

When an ad call includes `EXCLUDE=category ID,category ID,category ID...`, any flight having any of the specified categories is excluded from consideration during the call. A `viewid` is not necessary. If a `viewid` is present, the normal advertiser-based exclusions that might already be present on the page are still applied, and the explicit list supplements them. The `EXCLUDE` categories are applied only in the ad call in which they appear; they do not automatically carry over to later ad calls.

Using `EXCLUDEP=category ID,category ID,category ID...` allows exclusions to occur in the ad call in which it appears, as well as carry over into all subsequent calls with the same `viewid`. This exclusion happens even when those subsequent calls do not include the `EXCLUDE` or `EXCLUDEP` tags. When an ad call containing `EXCLUDEP` is received with no `viewid`, it behaves like the `EXCLUDE` token and affects only the ad call in which it appears.

When the exclusion tokens setting is enabled, the token `%%EXCLUSIONS%%` is available for substitution in creative content. The `EXCLUSIONS` token produces a comma-delimited list of category IDs that have served the `viewid` in the ad call. The list includes any categories associated with the flight served by that call and any `EXCLUDE` categories that are expressed in the call. If no `viewid` is in the call, the `%%EXCLUSIONS%%` token contains only the `EXCLUDE` categories that are expressed in the call. Contact SAS Technical Support to enable this setting.

SAS® 360 Match currently allows entry of categories containing commas, spaces, and other punctuation. However, categories containing spaces cannot be expressed in the `EXCLUDE` values, nor are they distinguishable when appearing in the `EXCLUSIONS` token values.

Maximum Request Length

The maximum size of a request is 32767 bytes. Anything longer is truncated. There are no additional limits on the path info, query string, or header portions of the ad call.

Video Ads

Ad call methods can be used to call VAST ads. Determine the VAST-compliant video player that you are using to make the ad request.

When you serve VAST video ads directly from SAS® 360 Match, use the following VAST creative format:

```
VAST version="2.0">
  <Ad id="aiMatchdirect_%%FCID%%_%%X_AD_SYSTEM%%">
    <InLine>
      <AdSystem>%%X_AD_SYSTEM%%</AdSystem>
      <AdTitle>%%X_AD_TITLE%%</AdTitle>
      <Description>%%X_AD_DESCRIPTION%%</Description>
      <Impression id="aimatch">
        <![CDATA[ %%BEACONURL%% ]]>
      </Impression>
      <Creatives>
        <Creative>
          <Linear>
           <Duration>%%X_DURATION%%</Duration>
           <TrackingEvents>
             <Tracking event="AdComplete">
               <![CDATA[ ]]>
             </Tracking>
           </TrackingEvents>
           <MediaFiles>
             <MediaFile delivery="progressive" bitrate="500"
             width="%%X_WIDTH%%" height="%%X_HEIGHT%%" type="%%X_VIDEO_TYPE%%">
               <![CDATA[ %%MEDIA%% ]]>
             </MediaFile>
           </MediaFiles>
          </Linear>
        </Creative>
      </Creatives>
    </InLine>
  </Ad>
</VAST>
```

The following data fields are available in the creative format: AD_SYSTEM, AD_TITLE, AD_DESCRIPTION, ERROR_URL, DURATION, and VIDEO_TYPE, which is a required field. Possible values for VIDEO_TYPE are video/x-flv or video/mp4.

The MIME type for this creative format is application/xml.

When you serve third-party video ads, use the following VAST wrapper creative format:

```
<VAST version="2.0">
  <Ad id="aiMatchwrapper_%%FCID%%_%%X_AD_SYSTEM%%">
    <Wrapper>
      <AdSystem>%%X_AD_SYSTEM%%</AdSystem>
      <VASTAdTagURI>
        <![CDATA[ %%X_AD_URL%% ]]>
      </VASTAdTagURI>
      <Impression>
        <![CDATA[ %%BEACONURL%% ]]>
      </Impression>
      <Creatives>
```

```
          <Creative AdID="%%FCID%%_aiMatch_creative_id">
            <Linear>
             <TrackingEvents>
               <Tracking event="AdComplete">
                 <![CDATA[ ]]>
               </Tracking>
             </TrackingEvents>
            </Linear>
          </Creative>
        </Creatives>
      </Wrapper>
    </Ad>
  </VAST>
```

The following data fields are available in the creative format:

- AD_SYSTEM: Define the advertiser or third-party system that the wrapper is pointing to.
- AD_URL: Specify the URL for the third-party VAST ad.

The MIME type for this creative format is application/xml.

Security Headers

If you need additional XSS or other security headers, you can have HTTP response headers set for every response from the ad server. The setting tells SAS® 360 Match which file to include for headers. The file can contain template variables that are substituted just as they are for template creatives or creative formats. Some headers might prevent inclusion of content served from SAS® 360 Match on third-party websites. Contact Technical Support at support@sas.com for more information.

Directive and Tag Aliasing

Engine directives and tag names can be aliased to maintain URL consistency or to circumvent ad-blocking software. Any engine directive and tag name can be aliased and be assigned multiple aliases.

Assigning multiple aliases allows for migration to new aliases over time without requiring all ad requests to adopt the new alias immediately.

For example, the `HSERVER` directive could be assigned an alias of `HTML`. A request using this alias might look like this:

```
https://shortname-ads.aimatch.com/{shortname}/HTML/site=x/area=y/size=728x90/
pos=top/random=345678
```

When an alias is defined, directives or tag names in responses from SAS® 360 Match will include aliases. For example, if the tag `SITE` is assigned an alias of `SECTION` and the directive `ADCLICK` is assigned an alias of `CLICKED`, the request might look like this:

```
<iframe src="https://shortname-ads.aimatch.com/{shortname}/HSERVER/
SECTION=x/area=y/size=728x90/pos=top/random=345678"></iframe>
```

The response would include:

```
<a href="https://shortname-ads.aimatch.com/{shortname}/CLICKED/SECTION=x/area=y/
size=728x90/pos=top/random=345678"><img src="#"></a>
```

Aliases are confined to requests and responses from SAS® 360 Match. Reporting data, targets, tokens, and supertag definitions still use the names that are not aliased.

To enable aliasing, contact SAS Technical Support.

Cross Origin Resource Sharing

Cross Origin Resource Sharing (CORS) policies define how a website controls reading responses or data access requests from other websites.

Trusted Domains

To protect against malicious attacks from external sources, websites should grant access only to trusted domains. The list of allowed external domains should be carefully selected and revalidated regularly to prevent unauthorized data access via third-party security breaches.

You can define a list of acceptable origin header values that enable CORS headers to be included in the response. This list should consist of domain names only, without the `http://` or `https://` prefixes. CORS headers are added only to requests with an Origin header matching one defined in the list. Contact SAS Technical support to configure this option.

Cross-Domain Applications

A CORS cross-domain policy allows a user to control whether a web client can handle data across domains. For more information, see https://www.adobe.com/devnet/adobe-media-server/articles/cross-domain-xml-for-streaming.html.

You can customize the body of the crossdomain.xml response by defining a custom creative format in the SAS® 360 Match user interface. The format must be named `crossdomain-xml`, with a hyphen instead of a period in the name. The content is entered either by as text or uploaded in a file. The MIME type must be application/xml.

When a call is made to SAS® 360 Match to fetch crossdomain.xml, and the customer has created a `crossdomain-xml` format, the text of that format is returned as the response.

If the customer has not defined a `crossdomain-xml` format, then the standard crossdomain.xml is returned, which is:

```
<cross-domain-policy>
<allow-access-from domain="*" to-ports="*" />
</cross-domain-policy>
```

Default Creative Format

Use the `DEFAULT=anydefaultname` tag to specify a default creative format to serve if an ad engine times out, where `anydefaultname` is the name for your default creative format. When an ad request includes this tag in the targeting path information and the ad engine takes too long to respond, the creative format named `anydefaultname` is selected to serve.

If the `anydefaultname` creative format is found, its content is returned as the response. If `anydefaultname` is not found, the ad server looks for a format named `default`. If the `default` creative format found, its content is returned. If `default` is not found, the default pixel is returned.

Define the default creative format with content that is appropriate for its context. For example, if the caller expects XML, the default creative format should contain XML. You can also include beacon calls to SAS® 360 Match to record when the default creative format is served and add the ability to handle clicks on the default content.

Different default creative formats can be defined with different sizes, depending on the context of the page in which they might appear. Ensure that there is a unique default creative format defined for each of these cases and reference the appropriate format name as the value for the `DEFAULT` tag.

NOTE: You can use any name in place of `anydefaultname` for your default creative format. However, you must use the name `default` for the creative format that SAS® 360 Match serves if your named format is not found or if no `DEFAULT` tag is in the ad call.

CNAMEs

Customers can obscure third-party requests like ad requests on their websites to make the content appear to originate from their domain. Canonical name, or CNAME, records allow a customer to use their first-party domain, such as ads.customer.com, for their ad request domain instead of the default sas.com domain.

A CNAME is a type of DNS record that maps, or aliases, one domain name to another. CNAMEs can also be used to circumvent ad blocking and privacy protection tools.

CNAMEs are supported for the ad request domain and the content distribution network, or CDN, domain. The default ad request domain is *shortname-ads*.aimatch.com, and the default CDN domain is content.aimatch.com.

When using a CNAME for the ad request domain, Amazon issues and manages a TLS certificate for the CNAME. For CDN CNAMEs, the CNAME is added to an Akamai SAN (Subject Alternative Name) TLS certificate.

To begin using CNAMEs, contact SAS Technical Support.

Spoofing Geotargeting Data

You can spoof geotargeting data using various parameters in an ad call. These parameters override certain values that are passed in via the HTTP request header, such as the IP address and cookie value. The parameters can be used to spoof geographic locations other than the actual location of the visitor, as determined by a lookup of the IP address to resolve its geographic location.

In the table below, the left column lists the geotargeting items. The middle column lists the tag name to use in SAS® 360 Match and in the ad call when you spoof geotargeting. Some targeting items, such as metro codes, are shown as codes rather than strings in the SAS® 360 Match target builder. Links to files that map codes to the values that they represent are in the third column.

Targeting Item	Targeting and Ad Call Tag	Designated Code
City	geo_city	City codes
Metro code (for U.S., Great Britain, and France only)	geo_metro_code	Metro codes
Country	geo_country	Country codes
Region (state or province)	geo_region	Region codes
IP address	geo_ip	
Postal code (ZIP code)	geo_zip_code_text	
Connection speed	geo_conn_speed	

The geotargeting lookup for a visitor's IP address occurs on a visitor's first request and remains in the system memory for the duration of the session. Therefore, delivery and troubleshooting can go awry if a visitor makes an initial ad request from their actual geographic location and then spoofs geotargeting tags during the same session. This change causes misleading results because SAS® 360 Match uses the geotargeting data from the initial request, not from the subsequent spoofing requests. To avoid this, you can add a unique cookie value for each request, thus guaranteeing a new session each time. The default cookie name in SAS® 360 Match is `mid`. Use the appropriate cookie name for your ad server domain if it is something other than the default.

In this example, a flight exists with a target of `state=AZ` for Arizona and the trafficker's IP address resolves to New York. To confirm that visitors from Arizona are receiving the flight, the trafficker would spoof their location. The troubleshooting ad call would look something like this:

```
http://view.adserver.com/clientid/hserver/size=300x250/mid={some-
number}/geo_region=AZ
```

In the ad call above, `{some-number}` is a different value each time.

In SAS® 360 Match, you can configure these tags in **Targeting > Targets** and in **Traffic** > **Debug Ad Request**.

Using the POST Directive

You can use the POST directive to communicate with the ad API. This mechanism enables BSERVER and SETSV calls to include more information than can be contained in a URL, which is limited to 2,000 characters.

The POST body must be in JSON format. Any type of request can be issued using POST. If an element named path_info is found, it is extracted and appended to the path information in the URL, automatically adding a slash between them if needed.

The resulting combined path information is used in the same way as the path information data that is in the URL. Therefore, any type of request is supported.

For example, a POST to https://*shortname-ads*.aimatch.com/tenant/hserver/mid=1234 with `{"path_info": "site=abc/area=xyz"}` is treated like a GET from https://*shortname-ads*.aimatch.com/tenant/hserver/mid=1234/site=abc/area=xyz.

The POST data can include the entire path information, including tenant and type of ad call. For example, a POST to https://*shortname-ads*.aimatch.com with `{"path_info": "tenant/hserver/mid=1234/site=abc/area=xyz"}` provides the same results as the example above.

Any query string in the URL is unchanged while the path information is combined with POST data. For example, a POST to https://shortname-ads.aimatch.com?search=dog&result=10 with `{"path_info": "tenant/hserver/mid=1234/site=abc/area=xyz"}` generates https://shortname-ads.aimatch.com/tenant/hserver/mid=1234/site=abc/area=xyz?search=dog&result=10.

There is one important difference between POST calls and GET calls. When a GET call arrives without a `MID` cookie, Match redirects the call to itself with LSERVER as the directive and attempts to set a new `MID` cookie to establish the visitor's identity. POST calls cannot do this.

Therefore, make POST calls only when the value of the `MID` cookie (or other appropriate ID that has been configured for visitor identity) is included explicitly in either the call's URL or POST data. Without such an identifier, the request is processed using a phantom `MID` value.

Using the Proxy Directive

Use the `proxy` directive with a URL parameter in the API request to send a proxy URL in the request. The URL parameter must meet the following requirements:

- Must be the last parameter in the request.
- Can be URL-encoded or not.
- Must match one of the URL definitions in the Privacy Shield. If there is no match, the proxy request returns a 404 Not Found error.

For example, if the URL is `https://testURL.com/myGIF.gif`, the API request is `https://myserver.com/proxy/url=https://testURL.com/myGIF.gif`.

The proxy request does not establish or use a visitor session. The request does not use the MID cookie or other visitor identifiers. You cannot use visitor data in the proxy templates.

Other tags are also parsed from the proxy request and available to reference as tokens in the proxy request templates. Geo and device tags are not looked up. These tags are available only if they are provided explicitly in the request. You can use regular and targeted supertags.

If the proxy request includes `trace=1`, the following items are returned to the client in a text response:

- The complete original client request
- The proxy request that is produced by the proxy request template
- The raw proxy response
- The processed final response that is produced by the proxy response template

Ad Call Methods

iserver

returns a 302 RELOCATE response and the image source URL. This call is designed to be served to the `<img>` tag. For example, you could insert the `iserver` URL into the `src` attribute of the `<img>` tag. For example:

```
<img src="http://shortname-
ads.aimatch.com/{shortname}/iserver/site=x/area=y/viewid=438943/random=948576789/s
ize=728x90" width=728 height=90 border=0>
```

hserver

returns the text source of a creative with the tokens replaced as defined in SAS® 360 Match. This call is designed to be served to the `SRC` attribute of the `<iframe>` tag. For example:

```
<iframe src="http://shortname-
ads.aimatch.com/{shortname}/hserver/site=x/area=y/viewid=438943/random=1827473656/
size=728x90">
```

jserver

returns JavaScript that dynamically writes the advertisement into the web page that called it. This call is designed to be served to the `SRC` attribute of the `<script>` tag. For example:

```
<script src="http://shortname-ads.aimatch.com/{shortname}/jserver/site=x/area=y/
viewid=438943/random=102958674/size=728x90">
```

bserver

is a special-purpose ad call that returns a bundle of advertisements with a single call. The system parses out the ad call parameters for each ad in the bundle, and then returns JavaScript code that contains the HTML code necessary for all the ads. This code presents the ads as a group of numbered JavaScript variable strings. For example:

```
<script src="http://shortname-ads.aimatch.com/{shortname}/bserver/ball/site=x/
area=y/viewid=43892143/random=09874756/b1/size=728x90/b2/size=728x90">
```

bserverj

is an ad request directive that operates similarly to a `bserver` call except that the `bserverj` call returns a JSON array of creatives in the response.

dserver

requests zero or more ads whose combined duration is less than or equal to the value specified in the `DURATION` tag. As with other directives involving the `DURATION` tag, only creatives having their duration field set are eligible for serving. To fulfill serving this directive, the Engine internally makes a series of ad selections. Each of the selections starts at the top of the ad queue. Tier and flight settings operate as usual on every selection. After each selection, it reduces the remaining duration by the duration value of the creative just selected and makes another selection attempt. The series of selections ends when a default ad is selected. The default is not returned in the response.

SAS® 360 Match uses the same tags passed in with the ad call on every selection. One additional tag (`ADPOS`) is synthesized on each selection and its value varies with the order of the selection. The first selection is made with `ADPOS=01`, the second with `ADPOS=LAST`, and the remaining ones with `ADPOS=02, 03, 04`, and so on. In this way, ads can be targeted to specified selection positions. Because SAS® 360 Match cannot know ahead of time how many ads will be selected, it forces the second selection to be tagged as `LAST`, which then allows for targeting this special position.

The system does not presently create the `ADPOS` tag itself or its values, so they are not available in the user interface. If you want to use `ADPOS`, you must manually create it in the user interface with the values of `01, 02`, and so on, and `LAST`, to match the values synthesized by SAS® 360 Match.

A `viewid` must be specified if certain ad selection features are desired, such as competitive exclusion, eliminate duplicates, companion ads, and so on, as usual. That is, no `viewid` is synthesized automatically for the request bundle.

The response consists of a simple concatenation of all creatives. The second ad selected (tagged as `LAST`) is forced to be at the end of the response. The MIME type provided is that of the first creative. If no ads are selected, the response body is empty. No default is returned.

tserver

returns the text source of a creative with the tokens replaced as defined in SAS® 360 Match. The MIME type of the ad response is defined by the creative format. This call is most often used with video players when serving VAST XML content. For example:

```
http://crtl.aimatch.com/{shortname}/tserver/site=x/area=y/viewid=438943/random=390
583034/size=728x90
```

Building an Ad Call Using Client-Side JavaScript

Because modern browsers ordinarily parse JavaScript, you can create dynamic ad calls on the page for each of the methods described in the **Ad Call Methods** section above. All examples here are shown using client-side JavaScript to build the ad call dynamically. Comparable ad calls can be created using server-side dynamic solutions such as ASP, ColdFusion, PHP, and so on.

iserver Ad Call

The iserver ad call method is used to return a 302 RELOCATE response and a relocation URL to an `<img src=>` element on the page. This ad call is often used in email newsletters as most email readers do not accept the hserver | jserver | bserver methods.

Strengths

- Simplicity
- Enables ad delivery into email newsletters

Weaknesses

- Does not support rich media creatives; supports static image creatives only.

- Does not allow the insertion of a unique identification number for the creative, or FCID. The FCID positively correlates clicks into the `<a href=>` URL at any time after the creative has been served. Omitting an FCID can result in incorrect click-throughs. Email newsletter content management systems typically allow the use of variable tokens such that a unique mid value can be placed in the `<img src=>` and `<a href=>` URLs.

Example

Notes:

- Replace `{path to your ad server}` with the correct ad server domain URL, including your specific `{clientid}` value.
- Replace `{ad call targeting}` with the correct targeting parameters for each ad instance.
- Replace the `width=` and `height=` values with the correct sizes.

```
<a href="http://{path to your ad server}/adclick/{ad call targeting}"><img
src="http://{path to your ad server}/iserver/{ad call targeting}" width={width}
height={height} border=0></a>
```

Email Example

Notes:

- To avoid encoding issues with some email readers, insert the targeting values in the query string using an ampersand (&) as the delimiter between each key=value pair.
- The publisher's content management system that generates the email needs to generate two sets of values for SAS® 360 Match to accurately pair the ad image with the click-through. These values are `mid={value}` and `pid={value}`.
- For multiple ad calls in each newsletter, the `mid` value should remain the same for all ad calls, and the `pid` value should be unique for each set of `<a href>` and `<img src>` tags.

```
<a href="http://{path to your ad server}/adclick?site=x&area=y&size=728x90
&mid=1234&pid=5678"><img src="http://{path to your ad server}/iserver?
site=x&area=y&size=728x90&mid=1234&pid=5678" width={width} height={height}
border=0></a>
```

hserver Ad Call

The hserver ad call method is used to return content to an `<iframe src>` element on the page. The iframe is defined in the parent document but is configured to reference SAS® 360 Match (with the appropriate ad call parameters) for its content. The system returns the advertisement as a complete document.

Strengths

- Enables the insertion of rich media (file-based advertisements).

- Not likely to delay the rendering of the page content because modern browsers usually render iframes asynchronously.
- Enables the insertion of a unique identification number (FCID) that positively correlates clicks at any time after the creative has been served. There is no expiration on click functionality.
- Supports all creatives that can function in the browser.

Weaknesses

- Does not support creatives that resize themselves or modify their position on the page without additional coding on the page.
- Can cause issues with creative that are designed to open in the parent page and not as a new document.

Example

Notes:

- Replace `{path to your ad server}` with the correct ad server domain URL, including your specific `{clientid}` value.
- Replace `{ad call targeting}` with the correct targeting parameters for each ad instance.
- Replace the iframe `width=` and `height=` values with the correct sizes.
- If you are using Minify JavaScript remove all HTML comments beginning with `<!—`.

```
<script type="text/javascript" language="JavaScript">
<!-- Hide from old browsers
// Modify to reflect site specifics
ad server = "http://{path to your ad server}";
target = "/{ad call targeting}";

// Cache-busting and view ID values
random = Math.round(Math.random() * 100000000);
if (!pageNum) var pageNum = Math.round(Math.random() * 100000000);

document.write('<iframe src="'+ad server+'/hserver/random='+random + target +
'/viewid=' + pageNum + '"');
document.write(' noresize scrolling=no hspace=0 vspace=0 frameborder=0
marginheight=0 marginwidth=0 width ={width} height={height}
allowTransparency="true">');
document.write('</iframe>');
// End Hide -->
</script>
```

jserver Ad Call

The jserver ad call method is used to return content to a `<script src>` element on the page. The `<script src>` ad call with is made and the system returns the selected ad wrapped in JavaScript, which is then written dynamically into the page and parsed.

Strengths

- Enables the insertion of rich media (file-based advertisements).
- Enables the insertion of a unique identification number (FCID) that positively correlates clicks at any time after the creative has been served. There is no expiration on click functionality.
- Supports all creatives that can function in the browser.

Weaknesses

- JavaScript could delay the rendering of the rest of the content if the response of the ad server is degraded. See Building an Ad Call Using Krux PostScribe for more information.
- Poorly written or malformed advertisements might interfere with or (in extreme cases) break page content.

Example

Notes:

- Replace `{path to your ad server}` with the correct ad server domain URL, including your specific `{clientid}` value.
- Replace `{ad call targeting}` with the correct targeting parameters for each ad instance.
- Replace the iframe `width=` and `height=` values with the correct sizes.
- If you are using Minify JavaScript, remove all HTML comments beginning with `<!–`.
- Creative content URLs that are protocol neutral (meaning that the URL begins with `//` rather than `http:` or `https:`) do not render because the URL after the `//` is treated as a comment by the browser's JavaScript parser. Prefacing `//` with the `%HTTP%` token prepends the ad request URL's protocol, whether it is `http:` or `https:` so that the URL is properly parsed.
- The jserver method wraps each line of code with document.write() or document.writeln(). If required, this can be suppressed for one or more lines of the creative's code by wrapping the statements with '`<!--Begin JSERVER Skip--> {code} <!--End JSERVER Skip-->`'.

```
<script type="text/javascript" language="JavaScript">
<!-- Hide from old browsers
// Modify to reflect site specifics
ad server = "http://{path to your ad server}";
target = "/{ad call targeting}";

// Cache-busting and view ID values
random = Math.round(Math.random() * 100000000);
if (!pageNum) var pageNum = Math.round(Math.random() * 100000000);
```

```
document.write('<scr'); document.write('ipt src="' + ad server +
'/jserver/random=' + random + target + "/viewid=" + pageNum + '">');
document.write('</scr');
document.write('ipt>');
// End Hide -->
</script>
```

bserver Ad Call

The bserver ad call is the most efficient ad call. It has the smallest code footprint for multiple ad calls on a page and requires only one request to the system to get all ad content for the page. The single request aspect of the ad call also eliminates a race condition that can occur when multiple ad calls are made from the page simultaneously. A race condition can impair companion ad serving or sequential ad delivery.

The ad call construction shown below is required on the page only once, although you can use multiple bserver calls if you want to use multiple `viewid` values on a single page. The code to render each ad (also shown below) is inserted at each ad position on the page.

By default, bserver ad requests leave comments and newline characters in place so that text-based creatives appear as they are designed. If you want bserver ad calls to automatically remove comments and newline characters, contact SAS Technical Support.

Strengths

- Most efficient method of delivering multiple ads to a page.
- Enables the insertion of rich media (file-based advertisements).
- Enables the insertion of a unique identification number (FCID). The FCID positively correlates clicks at any time after the creative has been served, so there is no expiration on click functionality.
- Prevents a race condition from occurring.

Weaknesses

- JavaScript could delay the rendering of the rest of the content if the response of the ad server is degraded. See Building an Ad Call Using Krux PostScribe for more information.
- Poorly written or malformed advertisement code might interfere with or (in extreme cases) break page content.

Example

Notes:

- Replace `{path to your ad server}` with the correct ad server domain URL, including your specific `{clientid}` value.
- Replace `{ad call targeting}` with the correct targeting parameters for each ad instance.
- If you are using Minify JavaScript, remove all HTML comments that begin with `<!–`.

- Creative content URLs that are protocol neutral (meaning that the URL begins with `//` rather than `http:` or `https:`) do not render because the URL after the `//` is treated as a comment by the browser's JavaScript parser. Prefacing `//` with the `%HTTP%` token prepends the ad request URL's protocol, whether it is `http:` or `https:` so that the URL is properly parsed.
- Add the `adx` JavaScript variable and its accompanying `/bx/` string for each ad instance on the page, replacing the `x` with an incremented number.
- For each `adx` variable created, add an `adx +` string in the section of JavaScript, which builds the `script src` bserver ad call (shown below).
- Edit the rendering code to replace `x` with the correct `/bx/` value that was referenced in the ad call.

  Note: The initial value of `x` cannot be zero (0).

- The resulting bserver URL path information (the ad call request itself, not the JavaScript used to create it) cannot exceed 2048 characters.

- `/ball/` strings are appended to each `/bx/` ad request section of the bserver URL. This enables you to insert global ad call data into each ad request while defining it only once.

- Typically, this would be used for random and viewed values, but it can also include targeting tags (for example, if all ad calls on the page use the same `SITE` and `AREA` values). The `/ball/` string should appear after `bserver` in the ad call, but before any `/bx/` strings. If the ad call is not set up correctly, the system will not fail, but it might include extraneous strings in the ad requests.

- If the number of `document.writeln(bx)` calls on the page do not match the number of `/bx/` ads defined in the bserver `script src` call, the impressions are logged, even though the ads are not rendered. Make sure that the `bx()` calls match the `/bx/` ads defined in the bserver `script src` call.

- You can add multiple bserver calls can on a page but do not duplicate `/bx/` values between the calls.

- The `/ball/` string contains a `/random=` value that is appended to each `/bx/` ad request. Therefore, multiple ad requests to the same third-party ad server within the same bserver ad call contain identical `/random=` cache-busting values. The identical values might affect the third-party ad server's impression counts. One alternative is to use multiple bserver ad calls on the page.

- The ad call below uses the `aimRnd` variable value for both the `/random=` and `/viewid=` values.

- Rich media file-based ads should be stripped of all comment strings.

- There are known issues with Google ads when multiple Google ads are served to the same bserver call. Test the multiple Google ads with the bserver ad call in a test campaign and test environment before going live.

```
<head><script language="javascript" type="text/javascript"
><!--
```

```
var b1 = "";
var b2 = "";
var b3 = "";
var b4 = "";
//-->
</script>
<head>
<body>
<script type="text/javascript" language="Javascript">
<!-- Hide from old browsers
// Cache-busting and view ID value
var aimRnd = Math.round(Math.random() * 100000000);

// ad server URL
adserver = "http://{path to your ad server}/bserver";

// Ad tag targeting values that will be appended to each ad request section in the
bserver ad call
allAdTags = "/ball/random=" + aimRnd + "/viewid=" + aimRnd;

// Individual tags for each ad request - increment the adx variable name and the
'/bx/' parameter.
ad1 = "/b1/{ad call targeting}";
ad2 = "/b2/{ad call targeting}";
ad3 = "/b3/{ad call targeting}";
ad4 = "/b4/{ad call targeting}";

// bserver ad call – insert the adx variables
document.write('<scr' + 'ipt src="' + adserver + allAdTags + ad1 + ad2 + ad3 + ad4
+ '?" type="text/JavaScript" language="JavaScript">');
document.write('</scr' + 'ipt>');
// End Hide -->
</script>
```

At the ad locations, insert the following JavaScript (edit “bx” to reference the appropriate ad by number):

```
<script type="text/javascript" language="JavaScript">document.writeln(bx);
```

Building an Ad Call for Email Delivery

Here is an example of using an iserver ad call method to build an ad call for email delivery.

iserver Ad Call

The iserver ad call method is used to return a 302 RELOCATE response and a relocation URL to an `<img src=>` element on the page. This ad call is often used in email newsletters as most email readers do not accept the hserver | jserver | bserver methods.

Strengths

- Simplicity
- Enables ad delivery into email newsletters

Weaknesses

- Does not support rich media creatives; supports static image creatives only.
- Does not allow the insertion of a unique identification number for the creative ("FCID"). The FCID positively correlates clicks into the `<a href=>` URL at any time after the creative has been served. Omitting an FCID can result in incorrect click throughs. Content management systems for email newsletters typically allow the use of variable tokens so that a unique `mid` value can be placed in the `<img src=>` and `<a href=>` URLs.

Example

Notes:

- Replace `{path to your ad server}` with the correct ad server domain URL, including your specific `{clientid}` value.
- Replace `{ad call targeting}` with the correct targeting parameters for each ad instance.
- Replace the `width=` and `height=` values with the correct sizes.

```
<a href="http://{path to your ad server}/adclick/{ad call targeting}"><img
src="http://{path to your ad server}/iserver/{ad call targeting}" width={width}
height={height} border=0>
```

Email Example

Notes:

- To avoid encoding issues with some email readers, insert the targeting values in the query string using an ampersand (&) as the delimiter between each key=value pair.
- The publisher's content management system that generates the email needs to generate two sets of values for SAS® 360 Match to accurately pair the ad image with the click-through. These values are `mid={value}` and `pid={value}`.
- For multiple ad calls in each newsletter, the `mid` value should remain the same for all ad calls, and the `pid` value should be unique for each set of `<a href>` and `<img src>` tags.

```
<a href="http://{path to your ad server}/{shortname}/adclick?site=x&area=y&
size=728x90&mid=1234&pid=5678"><img src="http://{path to your ad server}/
{shortname}/iserver?site=x&area=y&size=728x90&mid=1234&pid=5678" width={width}
height={height} border=0></a>
```

Building an Ad Call Using Krux PostScribe

Remote scripts, especially ads, can block a page from doing anything else while they load. They can contribute to load times, which affects your bottom line. *Asynchronous* ads do not block the page and can be delivered after core content. However, asynchronous ads might contain calls to `document.write`, which expects to be handled synchronously.

PostScribe enables you to deliver a synchronous ad asynchronously without modifying the ad code. PostScribe uses DOM Proxies, which is a way to ensure that content is written similar to the way a browser would write content with `document.write/innerHTML`. Therefore, it behaves as the browser would, without complex parsing or hacks.

iframe Example

The following example uses iframe and merely demonstrates a simple way to use the PostScribe functionality. Feel free to use other patterns to create your ad call.

Notes:

- `aimRnd` is defined as a random number to prevent browser caching.
- `ad server` is defined as the base URL of all the request on the page. Replace `ad server` to match your account specifics.
- Replace `{path to your ad server}` with the correct ad server domain URL, including your specific `{clientid}` value.
- The variable `allAdTags` is defined as a variable to hold the common parameters that all the ad requests on this page use.
- In this example, the full ad call DOM object is built as an iframe and stored as a string in a variable. Previously declared variables are used while building the string.
- At each place in the content where you want an ad, use a `div` tag as the container object. Inside the tag is where the PostScribe library is called, the `div id` is passed, and the ad call identified. Ensure that the `div id` matches the first parameter in the PostScript call.
- You can download `htmlParser.js` and `postscribe.js` from the krux/postscribe GitHub site at https://github.com/krux/postscribe.

```
<html>
<head>
<script src="http://{path to your ad server}/path/to/htmlParser.js"></script>
<script src="http://{ path to your ad server}/path/to/postscribe.js"></script>
```

```
<script>
var aimRnd = Math.round(Math.random() * 100000000);

// ad server URL
adserver = "http://{your domain}/path/to/hserver";

// Ad tag targeting values that will be appended to each ad request section in the
bserver ad call

allAdTags = "site=testpages/area=async/random=" + aimRnd + "/viewid=" + aimRnd;
```

```
b1 = '<iframe src="' + adserver + '/' + allAdTags + '/size=728x90?" width="728"
height="90" scrolling="no"></iframe>';
b2 = '<iframe src="' + adserver + '/' + allAdTags + '/size=300x250?" width="300"
height="250" scrolling="no"></iframe>';
```

```
b3 = '<iframe src="' + adserver + '/' + allAdTags + '/size=160x600?" width="160"
height="600" scrolling="no"></iframe>';
b4 = '<iframe src="' + adserver + '/' + allAdTags + '/size=728x90?" width="728"
height="90" scrolling="no"></iframe>';
</script>
</head>
<body>
```

```
<div id='ad1'>
<h5>Advertisement</h5>
</br>
<script language="javascript" type="text/javascript">
   postscribe("#ad1", b1);
</script>

</div><div id='ad2'>
<h5>Advertisement</h5>
</br>
<script language="javascript" type="text/javascript">
   postscribe("#ad2", b2);
</script>
</div>

<div id='ad3'>
<h5>Advertisement</h5>
</br>
<script language="javascript" type="text/javascript">
   postscribe("#ad3", b3);
</script>
</div>

<div id='ad4'>
<h5>Advertisement</h5>
</br>
<script language="javascript" type="text/javascript">
   postscribe("#ad4", b4);
</script>
</div>
</body>
</html>
```

You can enable PostScribe tracking when ads are delivered by PostScribe by using a 3-argument form of the postscribe call. The third argument is a callback, and must be (or included in) a call to SASIA.ViewTracker.trackAdViews.

For example:

```
postscribe('#[dom-id]', '<script src="[js ad call url]"></scr'+'ipt>',
SASIA.ViewTracker.trackAdViews);
```

SASIA.ViewTracker.trackAdViews is a function that, when called, notifies the tracker that one or more new ads are in the DOM (Document Object Model) for it to track (likely after the ready/load events). This callback must be included for each PostScribe ad call on the page.

The top-level view-track.js script still needs to be included directly on the page. It is a static file served over s3.

Counting Directives

Requests can be made to alter an FCID's impression, click, action, and view counts.

Count

A request to SAS® 360 Match with the `count` directive increments or decrements a metric for a specified FCID. For example:

```
http://<shortname>-ads.aimatch.com/<shortname>/count/FCID=123/act=1/inc=1<html>
```

The FCID parameter is required. A `count` request can also contain targeting path information. This associates the `count` request with the included tag values. The following optional parameters can be used with this directive:

act

specifies the type of metric to count. The possible values are:

Value	Metric type
1	Impression
2	Click (when specified, a click is counted but the request is not redirected to the FCID's click URL)
3	Action (see the SAS® 360 Match Advanced Features Guide for usage instructions)
4	View

When no `act` value is specified, a value of 1 is assumed.

inc

specifies the number of counts to increment for the given metric and FCID. A positive or negative value can be specified. When no `inc` value is specified, a value of 1 is assumed.

Adclick

A request to SAS® 360 Match with the `adclick` directive increments a click count for a specified FCID, then redirect the request to the creative's click URL, if one exists. For example, when SAS® 360 Match receives the following request:

```
http://<shortname>-ads.aimatch.com/<shortname>/adclick/FCID=123/
```

the click count for FCID 123 increases by 1. The response to the request is an HTTP 302 Redirect to the click URL specified for the creative. If no click URL is specified, the response is an HTTP 302 Redirect to the engine default image.

Adclick requests can also contain targeting path information that associate the click count with the included tag values. Adclick requests are automatically generated by using the `%%CLICKURL%%` token in a creative format.

The following optional parameter can be used with this directive:

relocate

redirects the visitor to a URL specified in the request. For example, when SAS® 360 Match receives the following request:

```
http://<shortname>-ads.aimatch.com/<shortname>/adclick/
FCID=123/relocate=http://www.sas.com
```

the click count for FCID 123 increases by 1. The response to the request is an HTTP 302 Redirect to http://www.sas.com. This parameter enables a third-party ad server to use its click URL as the value for this parameter to ensure that a click is counted by both parties.

The `%%PRECLICKURL%%` token generates an adclick URL that includes the `relocate=` parameter.

Mobile

SAS® 360 Match provides several ways to control and optimize serving ads to mobile devices.

SDKs

Any mobile application or device can request and serve ads delivered by SAS® 360 Match if it uses a supported request method. SAS® 360 Match provides SDKs to create UI elements that can render SAS® 360 Match ads in iOS and Android mobile applications.

The SAS® 360 Match SDK API provides functions that are similar to Apple's iAd Framework and Google's AdMob/DoubleClick API. The SDK supports version 2.0 of the MRAID specification for rich media advertisements for the Interactive Advertising Bureau, or IAB. MRAID-compliant ads can expand up to a full screen, provide two-part creatives, respond to changes in the visibility of the ad, control screen orientation, and engage with other device functions.

Contact SAS Technical Support to download the SAS® 360 Match SDK and related documentation.

AMP-Enabled Pages

Accelerated mobile pages, or AMP, is an open-source library developed by Google to reduce page load times. On AMP-enabled pages, an AMP ad tag can be used to generate an ad request to SAS® 360 Match.

```
<amp-ad
  width="<width>"
  height="<height>"
  type="sas"
  layout="fixed"
  data-customer-name="<short name>"
  data-ad-host="<ad request hostname>"
  data-size="<size>"
  data-area="<area>"
  data-site="<site>"
  data-tags='{"TAGNAME1":"TAGVALUE1","TAGNAME2":"TAGVALUE2"}'>
</amp-ad>
```

The AMP framework translates this tag into a jserver ad call inside an iframe element on the page.

The `type` and `layout` attributes are required by the framework, and their values are always `sas` and `fixed`.

The `width` and `height` attributes specify the size of the iframe element rendered on the page.

The required `data-customer-name` attribute contains the customer short name. The `data-ad-host` attribute sets the ad request host name. The `data-size`, `data-site`, and `data-area` attributes define the size, site, and area values, respectively, for the ad request to SAS® 360 Match. The `data-tags` attribute is a JSON Blob with keys and values that are tag names and tag values. Make sure the single and double quotation marks in this attribute match the example provided.

Spider Filtering

SAS® 360 Match uses definition files from the IAB to determine whether a visitor's request originates from a spider, crawler, or bot. The evaluation is based on the user agent string provided in the request. The user agent string must match an entry in the allowlist, and must not match an entry in the denylist, be considered a legitimate request. If the request does not pass this evaluation, ads are returned as normal in the response but no counting metrics are incremented.

Fraud Detection

SAS® 360 Match uses a behavioral model to detect fraudulent visitors that were not identified by spider filtering. This model is based on the frequency of a given visitor's requests compared to the overall number of requests for a customer. Furthermore, the requests are evaluated separately by type: impressions, or clicks, actions, or views.

A visitor's requests are identified by their unique `MID` value. By default, if a visitor's `MID` value appears in impression requests more than 100 times in the past 1,000 impression requests, the visitor is considered fraudulent. If a `MID` value appears in click, action, or view requests 10 or more times in the past 200 requests, the visitor is also considered fraudulent.

Once a visitor is considered fraudulent, any request regardless of type is no longer counted for that visitor. Ads are still delivered as normal in responses.

Contact SAS Technical Support to configure the threshold levels for fraud detection.

Elements of an Ad Request

You can use tags, directives, and parameters to build an ad request. The following tags, directives, and parameters are supported by SAS® 360 Match.

Tags

A tag is used as a key-value pair in targeting path information. Values are optional for all tags.

Tag	Definition
`area`	Area tag
`dev_<name>`	Device tags
`geo_<name>`	Geotargeting tags
`keyword`	Keyword tag
`latitude`	Latitude tag
`longitude`	Longitude tag
`site`	Site tag
`size`	Size tag

Directives

A directive is a command to execute.

Directive	Definition
`adclick`	Increments a click count for the specified or inferred FCID and redirects visitor to the click URL specified for the creative, if one exists.
`bserver`	Bundled ad request directive.
`bserverj`	Bundled ad request directive with response in JSON format.
`count`	Directive that increments or decrements a count of a metric as defined by the `act` and `inc` parameters.
`dserver`	Duration ad request directive.
`hserver`	Ad request directive that returns unaltered creative code with text/html MIME type.
`iserver`	Image ad request directive.
`jserver`	JavaScript ad request directive.
`lserver`	Used in conjunction with ad request directives to assign cookies to visitors with no cookies.
`saspb`	Passback directive.
`setid`	Provides a cookie named “EXTERNAL” in a response containing the value specified.
`setsv`	Manipulates the state vector data for a visitor.
`tserver`	Text ad request directive. Uses any custom MIME type specified on a creative format.

Parameters

A parameter defines additional information, or settings, for a directive. For more information about some of these parameters, see the SAS® 360 Match Advanced Features Guide.

Parameter	Definition	Value Required
act	Used in conjunction with the count parameter. Defines the type of metric.	Yes
actid	ID of custom action.	Use either actid or actname
actname	Name of custom action.	Use either actid or actname
advid	ID of advertiser for action tracking.	Yes
append	Appends the specified tag values to the state vector or cookie. Used with setsv.	Not applicable
duration	Duration in seconds.	Yes
event	Used with setsv. Specifies the name of an event.	Not applicable
exclude	Categories to be excluded in an ad request.	Yes
excludep	Categories to be excluded persistently in ad requests.	Yes
external	Used with setid. Specifies the cookie name.	Yes
fcid	Process the request for the specified flight creative ID.	Yes
flightid	Serve a flight creative from the specified flight ID. Used with an ad request directive.	Yes
inc	Increments a metric by the specified amount. Used with ACT parameter.	Optional, default value is 1
mergefrom	Merges two different state vector identities.	Not applicable
mid	Alphanumeric visitor identification string.	Yes
nocompanion	Restricts companion flights from serving to the ad request.	Not applicable
nolog	Prevents the request from being logged as an impression, click, action, or view.	Not applicable
pbfcid	Used in passbacks.	Yes
pid	Distinguishes multiple ad requests in an email. Used with the iserver ad request directive.	Yes
random	A randomly generated value used for cache busting.	Yes
supertag	Specifies the supertag value.	Yes
trace	Debug ad request output.	Yes
ttl	Time-to-live value. Used in multiple directives.	Yes
value	Specifies the cookie value. Used with setid.	Yes
viewid	A randomly generated value typically the same for all ad requests on a page that changes every page view or refresh.	Yes