APPENDIX - M3 FEED IMPLEMENTATION GUIDE

DOCUMENTATION HISTORY

V1.10	2014.09.16 Mirago

The M3 Feed is available in seven output types, with a unified set of input parameters for all requests, thus making it possible to easily switch between outputs.

- **JAVASCRIPT** (client-side specific response): returns Javascript source which dynamically builds M3 adverts
- **HTML**: returns a complete XHTML (transitional) document
- **HTML FRAGMENT**: returns a <DIV> node populated with M3 adverts
- **XML**: returns an XML document
- **JSON**: returns a JSON formatted data object
- **JSONP**: returns a padded JSON data object
- **VAST**: returns a VAST formatted XML response

GENERAL RECOMMENDATIONS

In most cases the client-side SCRIPT request is the lightest, simplest and most flexible approach. This also automatically supplies the User-Agent and Referrer headers.

For server-side integrations, especially where structural manipulation (layout, aggregation with other feeds, etc.) is required, XML requests with appropriate 2nd party XSLT transformations or similar is recommended.

For AJAX type requests JSONP is generally recommended where server-side proxy requests are possible using JSON, but note that a SCRIPT request may better fit most of the common AJAX use cases and is much simpler to implement.

INPUTS

The input parameters for all M3 outputs are now unified to a common set of data. Some outputs support additional parameters specific to their usage, but these will be safely ignored or shared as appropriate by other output types.

All parameters defined here are querystring values to be appended to the request URL.

Header	Data Type	Required?	Description
User-Agent	String	Yes[1]	The User-Agent of the browser. In general, client-side implementations (JSON, JSONP, HTML, SCRIPT) do this automatically; server-side implementations must set this header in the request to M3 accordingly.

Referer	String	Yes[1]	The URL of the page that will receive the adverts from the request. In general, client-side implementations (JSON, JSONP, HTML, SCRIPT) do this automatically, server-side implementations must set this header in the request to M3 accordingly.

1. In the unlikely event of your chosen programming language or environment not allow you to set request headers, these can be supplied using the 'ua=' and 'ref=' parameters. The data must be URL encoded if sent in this manner. Please try and use headers if at all possible because the use of parameters causes unnecessary logging and may impact performance as a result.

Parameter	**Data Type**	**Required?**	**Description**
sys	String	Yes[6]	The M3 System to query
a	Integer	Yes	Associate code, consult your M3 account manager
s	String	No	Sub-associate identifier
ip	IP Address	Yes	End user IP address
n	Integer	Yes	Number of results requested (1-15)
q	String	Yes [2]	Keyword query for PPC adverts
u	URL	Yes [2+3]	URL to be matched to the site table if 'restrict to known sites only' is set and also by the Rule system.
f	Bit	No	Filter for adult content (default = 0)
b	Bit	No[4]	Use backfill results if required (default = 1)
i	Bit	No	Use images if available (default = 0)
bp	Enum	No	Return click prices. See below.
t	URL	No[5]	Tracking URL
ih	Integer	No	Logo image height for text ad
iw	Integer	No	Logo image width for text ad

cpw	Integer	No [7]	Copy width for banner ads
cph	Integer	No [7]	Copy height for banner ads
mt	String	No [8]	Media Type - banner, standardtext, video, bullettext or mobile
z	String	No	Zone name

2. Both ‘q’ and ‘u’ can be supplied in the same request and adverts will be selected and ranked on their own merit.
3. If the value of ‘u’ is passed as the string ‘REF’ (case insensitive), M3 will use the HTTP referrer header of the request or the ref= parameter. Please see note 1 regarding the use of ref=.
4. ‘Backfill’ results are 3rd party results which help to fill advert requests.
5. See the tracking section for details.
6. ‘sys’ only required if using ‘feed.mirago.com’ or other non-specific domain but recommended for future compatibility.
7. Required for banners.
8. Depending upon the system’s configuration, specific types of adverts can be requested by supplying the media type parameter. If this parameter is not supplied, the advert server is free to choose any media type that is valid for the other parameters in the request.

Enumerated inputs

Some inputs have a range of possible values.

Click Prices (Parameter ‘bp’):

Value	Description
NONE or 0	Do not return click prices (default)
INHERIT or 1	Return click prices in the default currency of the feed in question
GBP	Return click prices in Sterling (estimated if not feed default)
EUR	Return click prices in Euros (estimated if not feed default)
USD	Return click prices in US Dollars (estimated if not feed default)
	N.B. All prices are in cents/pennies NOT Pounds/Euros/Dollars e.g. 1.52 is ONE POINT FIVE-TWO CENTS, NOT ONE DOLLAR FIFTY-TWO!

Additional inputs

Some output formats have specific parameters applicable to their output type.

For script requests:

Parameter	Data Type	Required?	Description
d	String	No	An element ID that will be the destination for content. This must exist in the mark-up of the page or the ads will be appended to the end of the document. (default = 'mirago')
x	Char	No	One of the values 'p', 't' or 'b' (default) corresponding to the html anchor element's target attribute values '_parent', '_top' and '_blank'

For html requests:

Parameter	Data Type	Required?	Description
c	URI	No	URI for a CSS file
x	Char	No	One of the values 'p', 't' or 'b' (default) corresponding to the html anchor element's target attribute values '_parent', '_top' and '_blank'

OUTPUTS

Client-side Consumption

SCRIPT

Overview	Provides a response in the form of a javascript library which will create an inline <DIV> populated with M3 adverts. Recommended for general client-side implementations as it is extremely light-weight in size and design footprint, but implicitly requires scripting to be enabled at the browser.
Request Path	`{MIRAGOURL}/feed.script.ashx`
Additional Inputs	None, but note that since this method will render inline html, it will implement any styling available on the page.
Implementation	Inline mark-up in the body of an HTML outputting document.

Example	<html>
	<head>
	<script>
	var
	mirago={w:window,d:document,load:function(u){with(this){var
	f=function(){onload()};if(w.addEventListener){addEventListener('load',f,false);}else
	if(w.attachEvent){attachEvent('onload',f);}(navigator.userAgent.search('Firefox')?load=setTimeout(function(){fetch(u)},10):fetch(u));}},fetch:function(u){with(this.d){s=createElement('script');s.type='text/javascript';s.src=u;getElementsByTagName('head')[0].appendChild(s);}},onload:function(){this.exe=function(){this.onload()};},exe:function(){},dat:null}
	
	mirago.load('http://www.mirago.co.uk/feed.script.ashx?a=1000&n=5&q=mortgage&i=0&x=b&d=myadverts);
	</script>
	</head>
	<body>
	
	</body>
	</html>

HTML

Overview	Provides a response in the form of a complete XHTML page, intended for use in an iframe. Recommended for client-side implementations where browser scripting is expected to be disabled, but not generally because of its relatively high size.
Request Path	{MIRAGOURL}/feed.html.ashx
Additional Inputs	CSS (key: 'c', value: URL to a CSS file)
Implementation	Inline mark-up in the body of an HTML outputting document.
Example	<iframe src='http://www.mirago.co.uk/feed.html.ashx?a=1000&q=mortgage&n=2' />

Server-side Consumption

XML

Overview	Provides a response in the form of an XML document. Recommended for general server-side consumption.
Request Path	{MIRAGOURL}/feed.xml.ashx

Additional Inputs	
Implementation	HTTP request

Example (C#)
 	
(N.B. the current object is an HTTPRequest from IIS)
	`XPathDocument doc = null;`
	`HttpWebRequest feedRequest =`
	`(HttpWebRequest)WebRequest.Create('http://www.mirago.co.uk/feed.xml.ashx?a=1000&q=mortgage&n=2');`
	`feedRequest.Accept = @'text/xml';`
	`feedRequest.UserAgent = this.Request.UserAgent;`
	`feedRequest.Referer = this.Request.Referer;`
	`using(HttpWebResponse response = (HttpWebResponse) feedRequest.GetResponse())`
	`{`
	`  using(MemoryStream ms = response.GetResponseStream())`
	`  {`
	`  doc = new XPathDocument(ms);`
	`  }`
	`}`

Example (PHP)
 	
(N.B. this does not show setting of User-Agent or Referer headers)
	`$parser = xml_parser_create('ISO-8859-1');`
	`xml_set_element_handler($parser, 'foo','bar');`
	`xml_set_character_data_handler($parser, 'baz');`
	`if (!($file = fopen('http://www.mirago.co.uk/feed.xml.ashx?a=1000&q=mortgage&n=2', 'r'))) {`
	`  die('Could not open $xml_feed for parsing!\n');`
	`}`
	`while ($data = fread ($file, 4096)) {`
	`  if (!xml_parse($parser, $data, feof($file)))`
	`  {`
	`  die(sprintf( 'XML error: %s at line %d\n\n',`
	`xml_error_string(xml_get_error_code($parser)),`
	`xml_get_current_line_number($parser)));`
	`  }`
	`}`
	`xml_parser_free($parser);`

JSON

Overview	Provides a response in the form of a JSON structure, intended for use by server-side proxy requests for client-side AJAX requests. Recommended for client-side AJAX implementations where proxy requests are technically possible due to parsing and light-weight size. Please note that unproxied javascript access will be denied by browser security models.
Request Path	`{MIRAGOURL}/feed.json.ashx`
Additional Inputs	

Implementation	Client AJAX request via server-side proxy
Example	Situational, but may look similar to an XML request above

FRAG

Overview	Provides a response in the form of a fragment of an XHTML page (with a <DIV> root), intended for direct server-side integration into the body of an HTML outputting document.
Request Path	`{MIRAGOURL}/feed.frag.ashx`
Additional Inputs	CSS (key: ‘c’, value: URL to a CSS file)
Implementation	Server side include into the body of an HTML response
Example	Situational, but may look similar to an XML request above

TRACKING

M3 allows partners to maintain their own click-tracking mechanism for their feeds, but the details of how this is implemented have changed with this version of the M3 Feed.

As before a 2nd party URL is accepted as an input parameter. Previously, upon an end-user click the user would be directed initially to a M3 handler which redirected to the 2nd party tracking URL which redirected to the advertiser destination. With this version the flow will instead direct the user to a M3 handler which will redirect to the advertiser destination *whilst simultaneously* creating a fire-and-forget HTTP request to the 2nd party tracking URL.

With this model, the end user receives a faster response and their browser remains in the control of M3 until it is handed to the advertiser destination. The partner receives the same click data and is no longer responsible for handing off the end-user request to the advertiser.

MARK-UP AND STYLING

Previous versions of the M3 feed have handled styling (where appropriate) with inline CSS and style values generated in the response based directly on request parameters. This leads to a slightly slower, heavier response with limited flexibility and control for the 2nd party.

The 2010 version of the M3Feed instead accepts a CSS parameter for HTML outputs which will be referenced by an HTML <link> in the response. This allows our partners complete control and flexibility of the styling of their adverts and helps to keep the performance high.

Element	**Complete CSS selector**
Mirago container	div#mirago
Advert container	div#mirago a.a

*N*th advert container	div#mirago a.a.n*N*
Advertiser logo container	div#mirago a.a div.i
Advertiser logo	div#mirago a.a div.i img
Advert text container	div#mirago a.a div.x
Advert title	div#mirago a.a div.x div.t
Advert description	div#mirago a.a div.x div.d
Advert URL	div#mirago a.a div.x div.u

Mirago default CSS will be included where requested with the following parameter values:

- 'css=banner'
- 'css=skyscraper'

Skyscraper CSS will be included where no other value is provided.

M3 response mark-up is written into a container element. In the case of script format requests this is usually provided on the page already, but for html format requests and script requests without a pre-existing container element it consists of the following div element:

```
<div id='mirago'>
```

This affords an opportunity to specify a background, border, padding or other effect for the whole response html.

Results

For ***feed*** requests, each ***advert*** result (adunit) will be output as another child-element of the target container, as below:

```
<a class='a n{index}' href='{clickurl}' rel='nofollow'>
 <div class='i'>
  <img src='{imageurl}' title='{imagetitle}' alt='{imagetitle}'/>
 </div>
 <div class='x'>
  <div class='t'>{title}</div>
  <div class='d'>{description}</div>
  <div class='u'>{displayurl}</div>
 </div>
</a>
```

Example CSS

A simple example CSS to style as banner adverts might look like this:

```
#mirago
{
        border: 1px solid blue;
        padding: 5px;
        font-family: helvetica, arial, sans-serif;
        font-size: 75%;
        overflow: auto;
}
#mirago img
{
        border: none;
}
#mirago .a
{
        width: 270px;
        height: 70px;
        margin: 5px;
        padding: 5px;
        border: 1px solid silver;
        display: block;
        text-decoration: none;
        display: block;
        overflow: hidden;
}
#mirago .a:hover
{
        background-color: #99ccff;
}
#mirago .a .i,
#mirago .a .x
{
        float: left;
}
#mirago .a .x
{
        width: 170px;
}
#mirago .a .t
{
        font-weight: 700;
        height: 20%;
}
#mirago .a .d
{
        overflow: hidden;
        height: 60%;
}
#mirago .a .u
{
        color: green;
        height: 20%;
}
```