OMNITURE. Test&Target.

Expression Targeters and Script Profile Parameters

Simple Javascript knowledge is required to effectively use Script Profile Parameters and Expression Targeters. This document serves as a quick reference to make you productive with this functionality in just a few minutes.

- *Script Profile Parameters* are found under the mboxes/profiles tab. You can write Javascript programs that return any Javascript type (String, integer, array, etc)
- *Expression Targeters* are found under the mboxes/targets tab. You can write JavaScript programs that return a boolean 'true' or 'false' value. You'll often reference to script profile parameters in these targeters.

Visit the Targeting Center in online help for more information.

Script Profile Parameter Examples

Name: *user.recency*

```
var dayInMillis = 3600 * 24 * 1000;
if (mbox.name == 'orderThankyouPage') {
  user.setLocal('lastPurchaseTime', new Date().getTime());
}
var lastPurchaseTime = user.getLocal('lastPurchaseTime');
if (lastPurchaseTime) {
  return ((new Date()).getTime()-lastPurchaseTime)/dayInMillis;
}
```

Creates a variable for day as measured in milliseconds. If the mbox name is *orderThankyouPage*, set a local (invisible) user profile attribute named *lastPurchaseTime* to be take the value of the current date and time. The value of last purchase time is read, and if is defined, we return the time that has passed since the last purchase time, divided by the number of milliseconds in a day (which results in the number of days since the last purchase).

Name: *user.frequency*

```
var frequency = user.get('frequency') || 0;
if (mbox.name == 'orderThankyouPage') {
   return frequency + 1;
}
```

Creates a variable called *frequency*, initializing it to either the previous value or 0, if there was no previous value. If the mbox name is *orderThankyouPage*, the incremented value is returned.

Name: *user.monetaryValue*

```
var monetaryValue = user.get('monetaryValue') || 0;
if (mbox.name == 'orderThankyouPage') {
   return monetaryValue + parseInt(mbox.param('orderTotal'));
}
```

Creates a variable called *monetaryValue*, looking up the current value for a given visitor (or set to 0 if there was no previous value). If the mbox name is *orderThankyouPage*, new monetary value is returned by adding the previous one and the value of the orderTotal parameter passed to the mbox.

Expression Targeter Examples

```
return (new Date()).getHours() > 12;
```

Will match during any time after midnight and after noon (EST). You might use this in conjunction with

```
return (new Date()).getHours() <= 12
```

to deliver two different branches – one before and one after noon every day.

Note: Since the javascript is executed server-side, the date and time will be Eastern Time.

```
return (new Date()).getDay()== 0 || (new Date()).getDay()==6;
```

Will match on weekends (in server time Sunday is 0 and Saturday is 6). You could use this in conjunction with

```
return (new Date()).getDay()!=0 && (new Date()).getDay()!=6
```

which will match any day that is not Saturday AND not Sunday (any weekday, server time). This would enable delivery of specialized content on weekdays versus weekends.

```
return profile.get('gender') == 'male';
```

This matches any user who has an associated profile variable ‘gender’ which has its value set to ‘male’ in the mbox that uses the profile as the parameter. For example, an online dating website might use something similar to this in conjunction with

```
return profile.get('gender') == 'female';
```

to deliver different content to users whose profile data indicates their gender.

```
return landing.referrer.param('q').toLowerCase() == 'lpo' &&
       landing.referrer.domain.match(/google\.com/) != null;
```

Matches searches for 'LPO' from *google.com*. This expression is checking that the referring URL's query parameter 'q' has the value 'lpo' (effectively ignoring case) AND that the referring domain is google.com. The check for google is done by doing a regular expression comparison. A string's match() method works in the following way: If no match is made, a null is returned, an array of matches is returned otherwise.

```
return input.matches(input.extract(
 '.*yahoo\.[a-z\.]+/search.*[\?&]{1}p=([^&]+).*',
landing.referrer.url, 'EUC-JP'),
input.extract('.*yahoo\.[az\.]+/search.*[\?&]{1}p=([^&]+).*',
landing.referrer.url));
```

From the given string (landing.referrer.url) the section from regexp will be extracted. If encoding is specified (EUC-JP), the extracted section will be decoded. Extracted/decoded strings will be matched against "input" in accordance to matcher type defined on the Campaign Edit page (contains, does not contain, etc).

Objects and Methods

Shared

The following properties and methods may be referenced by both expression targeters and script profile parameters:

page.url	The current URL
page.protocol	The protocol used for the page (*http*, *https*)
page.domain	The current URL domain (everything before the first slash). For example, the bold text in http://**www.acme.com**/categories/men_jeans?color=blue&size=small
page.query	The query string for the current page. Everything after the '?'. For example, the bold text in http://www.acme.com/categories/mens_jeans?**color=blue&size=small**

page.param('<par_name>')	The value of the parameter indicated by <par_name>. If your current URL is Google's search page and you had inputted page.param('hl'), you would get "en" for the URL http://www.google.com/search?hl=en&q=what+is+asdf&btnG=Google+Search
page.referrer	The same set of operations as above apply for referrer and landing (in case of expression targeters) (i.e. referrer.url will be the url address of the referrer)
landing.url, landing.protocol, landing.query, landing.param	Similar to that of page, but for the landing page.
mbox.name	The active mbox's name.
mbox.param('<par_name>')	An mbox parameter by the given name in the active mbox.
profile.get('<par_name>')	The client-created user profile parameter by the name <par_name>. For example, if the user sets a profile parameter named "gender", the value can be extracted using "profile.gender". Returns the value of the "profile.<par_name>" set for the current visitor, returns null if no value has been set.
user.get('<par_name>')	Returns the value of the "user.<par_name>" set for the current visitor, returns null if no value has been set.
user.categoryAffinity	Returns the name of the best category
user.categoryAffinities	Returns an array with the best categories
user.isFirstSession	Returns true if it's the visitor's first session
user.browser	Returns the user-agent in the HTTP header. As an example, you can create an expression target to target Safari users only: `if (user.browser != null && user.browser.indexOf('Safari') != -1) { return true; }`

Expression Targeters only

The following properties and methods may only be referenced by expression targeters:

input.match(regexp, str)	returns true if *str* matches the *regexp*. For example: input.match(".*(productId=([0-9]+)).*", landing.url)
input.extract(regexp, var, encoding)	the last parameter (*encoding*) is optional. Extracts a substring from *var* accordingly with *regexp*. If *encoding* is specified, decodes the result with the given encoding. For example: input.extract("http://.*\&query=([^&]+).*", landing.url, "EUC-JP")
input.matches(stringToMatch1, stringToMatch2, ...)	*input* is an *Offermatica* class encapsulating an array of one or more strings, the strings are entered at the *Campaign Edit* page when using the targeter in question. In the example below, from given string (landing.referrer.url) a part from regexp will be extracted If encoding is specified, the extracted part will be decoded extracted/decoded strings will be matched against "input" in accordance to matcher type defined in Campaign Edit page (contains, does not contain, etc) input.matches(input.extract('.*yahoo\.[a-z\.]+/search.*[\?&]{1}p=([^&]+).*', landing.referrer.url, 'EUC-JP'), input.extract('.*yahoo\.[a-z\.]+/search.*[\?&]{1}p=([^&]+).*', landing.referrer.url)) The example above allows us to enter Japan text in input in UTF-8 encoding, and we will be able to match the equivalent of that string in EUC-JP encoding.
input.contains(val1, val2, ...)	Returns true if input (array of strings, set on the Campaign Edit page) contains all of the specified string values.

Script Profile Parameter only

The following properties and methods may only be referenced by expression targeters:

user.setLocal('attribute', value)	Creates or updates an attribute with the specified value visible only to other profile attribute scripts.
user.getLocal('attribute'):	Returns the value for a specified attribute visible only in other profile attribute scripts.

Common Operators

All standard JavaScript operators are present and usable. JavaScript operators can be used on strings and numbers (as well as other data types). A quick briefing:

==	Indicates equality. Holds true when operands on either side are equal.
!=	Indicates inequality. Holds true when operands on either side are not equal.
<	The active mbox's name.
>	Indicates that the variable on the left is greater than the variable on the right. Will evaluate to false if the variables are equal.
<=	Same as '<' except if the variables are equal then it will evaluate to true.
>=	Same as '>' except if the variables are equal then it will evaluate to true.
&&	Logically "ANDs" the expressions to the left and right of it – is only true when both sides are true (false otherwise).
||	Logically "ORs" the expressions to the left and right of it – is only true if one of the sides is true (false otherwise).

//	Checks if source contains all elements from target boolean contains (Array source, Array target) // extracts substring from target (corresponding to regexp) and decodes it Array/*String*/ decode(String encoding, String regexp, String target)
	The feature also supports the use of: constant string values, grouping e.g. (condition1 || condition2) && condition3, regular expressions e.g. :
	`/[^a-z]$/.test(landing.referring.url)`