VERSION 2.5

Tracking and Form Capture Configuration Guide

Force24™
Marketing Automation

Contents

Last update: 5 April 2018

Script Installation

Insert the following script block just before the closing </body> tag on each page where tracking and web capture is required:

```
<!-- Start Force24 tracking -->
<script>
     !function(f,o,r,c,e,_2,_4){f.Force24Object=e,f[e]=f[e]||function(){
     (f[e].q=f[e].q||[]).push(arguments)},f[e].l=1*new Date,
     _2=o.createElement(r),_4=o.getElementsByTagName(r)[0],_2.async=1,
     _2.src=c,_4.parentNode.insertBefore(_2,_4)}(window,document,
     "script","//tracking1.force24.co.uk/tracking/V2/main.min.js","f24");

     /* Set clientId */
     f24("create", "00000000-0000-0000-0000-000000000000");
     /* Place custom commands here */
     f24("send", "pageview");
</script>
<!-- End Force24 tracking -->
```

Note: The GUID within the f24("create", ..) should be replaced with the clientId supplied by Force24.

Tracking Configuration

Standard Tracking
Standard tracking requires no further configuration, each page view will be tracked when the browser's load event is triggered.

Extra Tracking Parameters
Extra parameters can be added to the tracking information for a page view by calling the 'set' command using the following syntax:

```
// Set custom parameters
f24("set", "key", "value");

// Unset custom parameters
f24("set", "key", null);

// Reset all custom parameters
f24("reset");
```

Please see 'ecommerce' tracking for a working example.

Dynamic Tracking
Many websites today load content dynamically via AJAX without requiring a full page load for each "page". These can be logged using the following syntax:

```
// Send custom page view
f24("send", "pageview", "http://mysite.com/page");
```

Ecommerce Tracking
Our system tracks the ROI of your campaigns, to push this data across to our system utilise our set command as follows:

```
// Send ecommerce tracking
f24("set", "CheckOutAmount", "9.99");
```

Form Capture Configuration

Each form being captured on the website needs to have unique attributes to allow the Force24 system to differentiate it from any other forms being captured.

All attributes are optional and can be set to any values you wish, but enough detail should be added so that each form captured has a unique combination of these attributes.

Attributes captured to the Force24 system on form submission are:

- **PagePath** – This is the path to the current page relative to the site root
- **id** – This is the standard ID attribute of the form element
- **name** – This is the standard Name attribute of the form element
- **data-f24name** – This is a custom attribute which can be added to the form element

Example form tag before any alterations:

```
<form class="myform" id="contact-form">
```

HTML forms (defined by a class)
By standard, our tracking script is configured to detect forms with the 'f24form' class applied:

```
<form class="myform f24form" id="contact-form" data-f24name="contact-form">
```

HTML forms (defined by your own selector)
You can also define the form selector using the following command:

```
f24("formSelector", "form#contact-form");
```

Note: This would be used in conjunction with the unique attributes as stated above.

Please notify Force24 with all the URLs of the forms you wish to setup, and we will take care of having these mapped to our system.

Self-Service form field mappings

You can specify your own mapping of form fields to person properties.
That way, visitor inputs on your forms will be mapped automatically to the correct person data.

Mapping is defined in a call to the *formMap* function. It takes an array of objects like this:

```
f24("formMap", [
  {
    url: /^url-for-this\/map.+/,
    selector: "form.subscription-form",
    meta: {id: "Lorem", name: "ipsum", f24name: "dolor"},
    fields: {
      FirstName: "givenname",
      EmailAddress: "email",
      Aux1: "{date} {time}",
      Aux2: ["date", "time"],
      Aux3: {key1: "date", key2: "time"},
      Aux6: "optin"
    },
    marketingList: "12345678-1234-1234-1234-123456789012",
    status: {
      optInEmail: "optin",
      optInSms: "optin-sms",
      optInDirectmail: "optin-directmail"
    },
    subscriptionPrefs: {
     "12345678-1234-1234-1234-123456789012": "checkbox1",
     "87654321-1234-1234-1234-123456789012": "checkbox2"
    }
  }
]);
```

formMap takes a JavaScript array with one or multiple form mapping configurations. Specify multiple config objects to target different forms on different pages.
They are specified as objects and have the following options:

- **url** – String or RegExp for part of the URL of the pages that this config should apply to.
- **selector** – See *formSelector* function.
- **meta** – Instead of adding meta data to your form HTML, specify them here.
- **fields** – The mapping object (see next page).
- **marketingList** – Useful for adding visitors to Force24 marketing lists.
- **status** – Reserved object with the keys *optInEmail*, *optInSms*, and *optInDirectmail*.
  The indicated form fields can be checkboxes or radios; sent values will be boolean.
- **subscriptionPrefs** – Subscription preference IDs with indicated form fields (see **status**).

Field mapping keys and configuration
In the *fields* object (see previous page for an example), each key must be a valid Force24 person property. The value is the name of an input, select or text area on your form.

Reference: Valid Force24 person properties
You can use the following properties as keys for the fields object:

- FirstName
- LastName
- MiddleName
- Salutation
- CompanyName
- JobTitle
- PhoneNumber
- HomePhoneNumber
- AddressLine1
- AddressLine2
- AddressLine3
- PostCode
- City
- County
- Country
- EmailAddress
- Aux1
- Aux2
- Aux3
- Aux4
- Aux5
- Aux6
- Aux7
- Aux8
- Aux9
- Aux10
- AuxDate1
- AuxDate2
- AuxDate3
- AuxDate4
- AuxDate5
- AuxDateTime1
- AuxDateTime2
- AuxDateTime3

Note: Please pay attention to the capitalisation.

Concatenating form field values
You can also specify multiple fields with curly braces, as array, or object (see sample code for *Aux1*, *Aux2*, and *Aux3* above). In this case, the data will be concatenated automatically into a plain string, an array or an object, respectively.

Configuration Extras

We have included extra parameters which can enhance your usage of our script.
Please consult our technical account management team for information around these options.

Anonymous data collection
If your account is configured for anonymous data collection, this command will allow the anonymous data to be passed back into our system.

```
f24("cookieAnonymous", true);
```

Disable cookie on form submission
By default, when a visitor submits a form, they are cookied so we can track their activity. Configuring the following command will prevent this.

```
f24("cookieOnFormSubmit", false);
```

Cookie domain
By default, we set the cookie to be assigned to the top level domain from common TLDs specified. In the event your domain isn't detected, you can specify the command as follows:

```
f24("cookieDomain", "example.com");
```

Automatic external link tracking
By enabling this command, our tracking script will attempt to detect and track external links and track these as visitor's exit to these sites.

```
f24("trackExternalLinks", true);
```

Manual form submission

This command can be used to send a submission of a form to us manually.

```
f24("send", "form", "form.my_form");
```

Manual form submission with custom data model

This command can be used to send a custom data array in the event of a form.

```
f24("send", "form", {f24name: "A", name: "B", id: "C", data: {key: "value"}});
```

Manual page tracking

This command can be used to send a manual page view to our system.

```
f24("send", "pageview", "http://mysite.com/page");
```

Form submission after capturing

Some websites don't submit their forms using "real" HTML, but instead process them entirely with JavaScript (Ajax). By default, the form capture script will track a form and then submit it to its original action. For Ajax forms, you can disable this using the following switch:

```
f24("formSubmitAfterTracking", false);
```

Resolving CMS restrictions

Some CMS don't allow to add custom attributes to the generated **<form>** tags.
In this case, you can use an alternative way to declare *f24name*, form ID and *class* attributes – using hidden fields:

```
<form class="generated-form" action="test-debug.php">
   <input type="text" name="email" value="john@doe.org">
   <input type="hidden" name="f24[f24name]" value="F24 The Name">
   <input type="hidden" name="f24[name]" value="form name">
   <input type="hidden" name="f24[id]" value="form ID">
   <button type="submit">Go</button>
</form>
```

In this case, you might want to adapt the form selector, since this form will be ignored by the tracking script – it has no *f24form* class:

```
// will capture all forms with hidden alternative tags
f24("formSelector", "form:has([name='f24[f24name]'])");

// alternative selector: will capture all forms
f24("formSelector", "form");
```

Alternative attribute for form input names
In case you are dealing with ASP.net or other forms where you cannot edit the *name* attribute of the *<input>*, *<select>* or *<textarea>* elements that you want to capture, you can add the alternative *data-f24-name* attribute. If this attribute is present, the actual name attribute on the form input field is ignored:

```
<input name="$c00somethingAutogeneratedByCMS"
   data-f24-name="email" type="email">
```

Javascript Callbacks

You can now use callbacks when using our tracking and form capture functions manually.

Manual page view with callback

```
f24("send", "pageview", "http://mysite.com/page", function () {
        /* Some logic */
});
```

Manual form submission with callback

This is an example with the manual form submission, however this can also be used with a custom data model.

```
f24("send", "form", "form.my_form",  function () {
        /* Some logic */
});
```