Boxalino Client to Server Tracking API

Technical Integration Manual

Document Version 1.0

August, 29th 2013

boxalino
THE INTELLIGENT ENGINE

Table of contents

1 Overview **2**

2 Simple Integration **3**
- 2.1 Sub-domain Tracking . . . 3
- 2.2 Identifying User Accounts . . . 4

3 Public API Reference **5**
- 3.1 void setAccount(accountId) . . . 7
- 3.2 void setCookieDomain(domain, path) . . . 8
- 3.3 void setOptions(kv) . . . 9
- 3.4 void setTrackerUrl(url) . . . 10
- 3.5 string getVisitor() . . . 11
- 3.6 string getSession() . . . 12
- 3.7 promise trackPageView(uri, page, variant, parameters) . . . 13
- 3.8 promise trackSearch(query, parameters) . . . 14
- 3.9 promise trackCategoryView(category, parameters) . . . 15
- 3.10 promise trackProductView(product, parameters) . . . 16
- 3.11 promise trackAddToBasket(product, quantity, price, currency, parameters) . . . 17
- 3.12 promise trackPurchase(total, currency, products, orderId, parameters) . . . 18
- 3.13 promise trackMouse(event, parameters) . . . 19
- 3.14 promise trackLink(event, parameters) . . . 20
- 3.15 promise trackInnerClick(event, url, parameters) . . . 21
- 3.16 promise trackOuterClick(event, url, parameters) . . . 22
- 3.17 promise trackGoal(name, orderId, productId, parameters) . . . 23
- 3.18 promise trackChoiceDisplay(choice, variant, visitor, scope, parameters) . . . 24
- 3.19 promise trackLogin(userId, parameters) . . . 25
- 3.20 promise trackChoiceClick(choice, parameters) . . . 26

1 Overview

Boxalino Client Tracking API is a pure javascript library without dependencies to trigger event notifications such as page views or purchases.

Events are sent from the client using minimal, hidden and transparent gif images. The javascript library associates a unique identifier with each visitor and collects information on the source of the traffic as well as a few selected demographic data (language, screen size, browser version and similar technical parameters).

The basic integration takes only a few minutes.

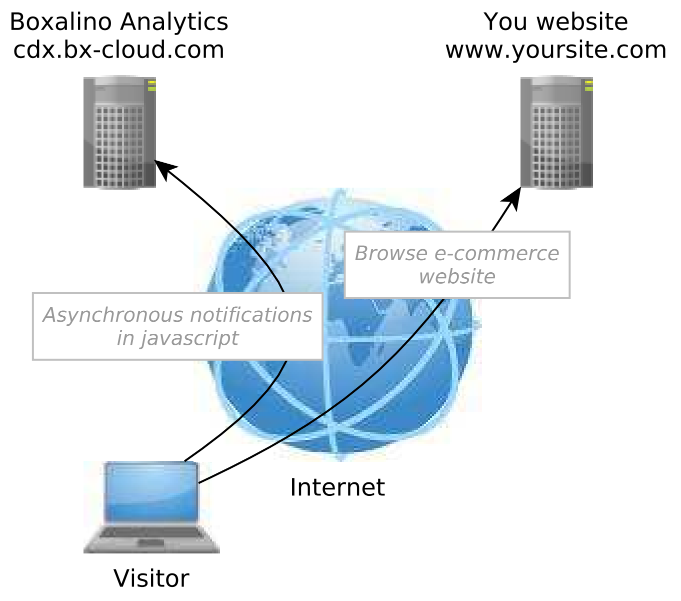

Figure 1: Interactions overview

2 Simple Integration

In order to integrate the library simply add the following html snippet in your page header:

```
<head>
    <script type="text/javascript">
        var _bxq = _bxq || [];

        _bxq.push(['setAccount', 'ACCOUNTID']);
        _bxq.push(['trackPageView']);

        (function(){
            var s = document.createElement('script');

            s.async = 1;
            s.src = '//cdn.bx-cloud.com/frontend/rc/js/ba.min.js';
            document.getElementsByTagName('head')[0].appendChild(s);
        })();
    </script>
</head>
```

This will send a page view event in the account 'ACCOUNTID'. Boxalino will provide your account identifier.

2.1 Sub-domain Tracking

In order to track your visitors properly across several sub-domains (ie. between promo.yoursite.com and www.yoursite.com) the following call must be done before any event tracking:

```
<script type="text/javascript">
    var _bxq = _bxq || [];

    _bxq.push(['setAccount', 'ACCOUNTID']);
    _bxq.push(['setCookieDomain', 'yoursite.com']);
    _bxq.push(['trackPageView']);

    (function(){
        var s = document.createElement('script');

        s.async = 1;
        s.src = '//cdn.bx-cloud.com/frontend/rc/js/ba.min.js';
        document.getElementsByTagName('head')[0].appendChild(s);
    })();
</script>
```

The cookie domain is simply the top-level name alone (ie. www.yoursite.com becomes yoursite.com). This code need to be installed on all active sub-domains.

2.2 Identifying User Accounts

When a visitor log-in your website as a registered user you can simply link the account identifier to the profile by making a call to this api:

```
<script type="text/javascript">
    var _bxq = _bxq || [];

    _bxq.push(['setAccount', 'ACCOUNTID']);
    _bxq.push(['setCookieDomain', 'yoursite.com']);
    _bxq.push(['trackPageView']);
    _bxq.push(['trackLogin', 'USERID']);

    (function(){
        var s = document.createElement('script');

        s.async = 1;
        s.src = '//cdn.bx-cloud.com/frontend/rc/js/ba.min.js';
        document.getElementsByTagName('head')[0].appendChild(s);
    })();
</script>
```

3 Public API Reference

Here are described in detail all the accessible methods of the API.

Each method can be called either using the javascript array syntax or a direct call.

Because the api is loaded asynchronously the _bxq object will not be accessible at page load. So the preferred way of calling tracking methods is through the array syntax.

As an example here are the two ways of triggering a page view event:

```
<script type="text/javascript">
    /* This use the array syntax (valid at any time).
       The first element in the array is the method name and the rest are method
           arguments. */
    _bxq.push(['trackPageView']);
    _bxq.push(['trackPageView', '/my/page.html']);

    /* Assuming that the _bxq object is initalized (after ba.js is loaded) it is
       possible to call the methods directly. */
    _bxq.trackPageView();
    _bxq.trackPageView('/my/page.html');
</script>
```

It is possible to be notified when the api is ready by adding a listener like so:

```
<script type="text/javascript">
    _bxq.push(['ready', function() {
        alert('_bxq object is fully ready!');
    }]);
</script>
```

With all tracking methods it is also possible to be notified when the tracking is done:

```
<script type="text/javascript">
    // array method: the callback is added at the end of the parameters
    _bxq.push(['trackPageView', function(status) {
        if (status) {
            alert('tracking is complete with success!');
        } else {
            alert('an error occurred in tracking!');
        }
    }]);

    // direct method: the callback is given to the returned promise
    var isdone = _bxq.trackPageView();

    isdone(function(status) {
        // tracking done
    });

    // direct method: without temporary variable
    _bxq.trackPageView()(function(status) {
        // tracking done
    });
</script>
```

3.1 void setAccount(accountId)

Set current account identifier for the tracking.

The effect is immediate and it returns nothing.

```
<script type="text/javascript">
    // async
    _bxq.push(['setAccount', 'ACCOUNTID']);

    // direct
    _bxq.setAccount('ACCOUNTID');
</script>
```

3.2 void setCookieDomain(domain, path)

Set domain and path for visitor cookies. If the domain is unspecified the current domain is taken. Path is optional and defaults to /.

The effect is immediate and it returns nothing.

```
<script type="text/javascript">
    // async
    _bxq.push(['setCookieDomain', 'mysite.com']);
    _bxq.push(['setCookieDomain', 'mysite.com', '/only/this/path']);

    // direct
    _bxq.setCookieDomain('mysite.com');
    _bxq.setCookieDomain('mysite.com', '/only/this/path');
</script>
```

3.3 void setOptions(kv)

Set custom tracking options such as cookie life time and timeouts. Valid options are:

1. timeout: timeout in milliseconds after which the event tracking is considered as failed (defaults to 1000 [ms]).
2. maxage: maximum age of the visitor cookie in milliseconds (defaults to one year).

The effect is immediate and it returns nothing.

```
<script type="text/javascript">
    // async
    _bxq.push(['setOptions', {timeout: 2*1000}]);

    // direct
    _bxq.setOptions({timeout: 2*1000});
</script>
```

3.4 void setTrackerUrl(url)

Changes the url of the tracker. All request will be sent to this url. This method should be used only if we want to run ba.js on customer's environment.

The effect is immediate and it returns nothing.

```
<script type="text/javascript">
    // async
    _bxq.push(['setTrackerUrl', 'URL']);

    // direct
    _bxq.setTrackerUrl('URL');
</script>
```

3.5 string getVisitor()

Get current visitor identifier.

It returns the visitor identifier as a string.

```
<script type="text/javascript">
    // only direct
    var visitorId = _bxq.getVisitor();
</script>
```

3.6 string getSession()

Get current session identifier.

It returns the session identifier as a string.

```
<script type="text/javascript">
    // only direct
    var sessionId = _bxq.getSession();
</script>
```

3.7 promise trackPageView(uri, page, variant, parameters)

Send a page view event to the tracker. The uri, page, variant and parameters are optional. If the uri is not given the current browser uri is taken. The key-value parameters can be used to append extra meta-data to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```
<script type="text/javascript">
    // async
    _bxq.push(['trackPageView']);
    _bxq.push(['trackPageView', '/custom/path.html']);

    // direct
    _bxq.trackPageView();
    _bxq.trackPageView('/custom/path.html');
</script>
```

3.8 promise trackSearch(query, parameters)

Send a search query event to the tracker. The query must be given and extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

Extra key-value parameters should be used to send filters used in a current search. Each filter needs to have prefix 'filter_'. There are several types of the filters which can be used:

- Prefix 'filter_' - simplest version of the filter, where given attribute has to be set to the proper value.
- Prefix 'filter_from_incl_' - lower boundary of the range filter (inclusive).
- Prefix 'filter_from_excl_' - lower boundary of the range filter (exclusive).
- Prefix 'filter_to_incl_' - upper boundary of the range filter (inclusive).
- Prefix 'filter_to_excl_' - upper boundary of the range filter (exclusive).
- Prefix 'filter_hrc_' - hierarchical filter, mostly used for categories (back slash '\' needs to be escaped too => "\\").

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```
<script type="text/javascript">
    // async
    _bxq.push(['trackSearch', 'text query']);

    // direct
    _bxq.trackSearch('text query');

    // direct with filters
    _bxq.trackSearch('text query', {
        filter_color: 'red', // single value filter
        filter_size: ['S', 'M'], // multiple values
        filter_from_incl_price: 10.0, // lower boundary of the range filter (inclusive)
        filter_to_excl_price: 100.0, // upper boundary of the range filter (exclusive)
        filter_hrc_categories: '/Parfums & Kosmetik/Gesichts- & Körperpflege/Lancome'
            // hierarchical filter
    });
</script>
```

3.9 promise trackCategoryView(category, parameters)

Send a category view/browse event to the tracker. The category identifier must be given and extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```
<script type="text/javascript">
    // async
    _bxq.push(['trackCategoryView', 'CATEGORYID']);

    // direct
    _bxq.trackCategoryView('CATEGORYID');
</script>
```

3.10 promise trackProductView(product, parameters)

Send a product view event to the tracker. The product identifier (i.e.: the product detail page id), not the SKU, must be given. This is important in the case you have several final article ids (SKU) which can be selected in one product detail page, the id must correspond to the id of a product detail page and not the selected final article number. Product identifiter is limited to be 50 characters long. Extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```
<script type="text/javascript">
    // async
    _bxq.push(['trackProductView', 'PRODUCTID']);

    // direct
    _bxq.trackProductView('PRODUCTID');
</script>
```

3.11 promise trackAddToBasket(product, quantity, price, currency, parameters)

Send an add-to-basket event to the tracker. The product identifier (i.e.: the product detail page id), not the SKU, must be given. This is important in the case you have several final article ids (SKU) which can be selected in one product detail page, the id must correspond to the id of a product detail page and not the selected final article number. The quantity, the price and the currency are optional. The quantity is an integer to specify how many times the product has been added to the basket. A negative quantity means that the product has been removed that many times. Product identifiter is limited to be 50 characters long. The price is given as a floating-point number and the currency is a string identifier (such as 'CHF'). Extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```
<script type="text/javascript">
    // async
    _bxq.push(['trackAddToBasket', 'PRODUCTID', 1, 59.90, 'CHF']);

    // direct
    _bxq.trackAddToBasket('PRODUCTID', 1, 59.90, 'CHF');
</script>
```

3.12 promise trackPurchase(total, currency, products, orderId, parameters)

Send a purchase event to the tracker. The total amount, the currency and the product list must be given. The total is given as a floating-point number and the currency is a string identifier (such as 'CHF'). The products array is a javascript array of object describing products. Each product must contain a "product" field containing the The product identifier (i.e.: the product detail page id), not the SKU, and optionally a "quantity" and "price" field. This is important in the case you have several final article ids (SKU) which can be selected in one product detail page, the id must correspond to the id of a product detail page and not the selected final article number. Product identifiter is limited to be 50 characters long. The orderId is a identifier of the order connected with the event. Extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```
<script type="text/javascript">
    // async
    _bxq.push([
        'trackPurchase',
        79.90,
        'CHF',
        [
            {product: 'PRODUCTID1', quantity: 1, price: 59.90},
            {product: 'PRODUCTID2', quantity: 2, price: 10.0}
        ],
        'ORDERID'
    ]);

    // direct
    _bxq.trackPurchase(
        79.90,
        'CHF',
        [
            {product: 'PRODUCTID1', quantity: 1, price: 59.90},
            {product: 'PRODUCTID2', quantity: 2, price: 10.0}
        ],
        'ORDERID'
    );
</script>
```

3.13 promise trackMouse(event, parameters)

Send a mouse position event to the tracker. The mouse state is extracted from the javascript dom event given as the first parameter. Extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```html
<a href="#" onclick="_bxq.trackMouse(event);">a link with tracking</a>
```

3.14 promise trackLink(event, parameters)

Send a link click event to the tracker. The first parameter is the javascript event triggered by the click on a link. The default click behavior which is to load the link url in the browser will be delayed until the event has been sent to the tracker. Extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```
<a href="#" onclick="_bxq.trackLink(event);">a link with tracking</a>
```

3.15 promise trackInnerClick(event, url, parameters)

Send a inner link click event to the tracker. The first parameter is the javascript event triggered by the click on a link. The second one is the url of the link. The default click behavior which is to load the link url in the browser will be delayed until the event has been sent to the tracker. Extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```html
<a href="#" onclick="_bxq.trackInnerClick(event, 'URL');">a link with tracking</a>
```

3.16 promise trackOuterClick(event, url, parameters)

Send a outer link click event to the tracker. The first parameter is the javascript event triggered by the click on a link. The second one is the url of the link. The default click behavior which is to load the link url in the browser will be delayed until the event has been sent to the tracker. Extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```
<a href="#" onclick="_bxq.trackOuterClick(event, 'URL');">a link with tracking</a>
```

3.17 promise trackGoal(name, orderId, productId, parameters)

Send a goal completion event to the tracker. The first parameter is the name of the completed goal. The orderId is a identifier of the order connected with the event. The productId is a identifier of the product connected with the event. Product identifiter is limited to be 50 characters long. Extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```
<script type="text/javascript">
    // async
    _bxq.push(['trackGoal', 'GOALID', 'ORDERID', 'PRODUCTID']);

    // direct
    _bxq.trackGoal('GOALID', 'ORDERID', 'PRODUCTID');
</script>
```

3.18 promise trackChoiceDisplay(choice, variant, visitor, scope, parameters)

Send a choice display event to the tracker. The choice, variant and visitor must be given. By default it is not needed to use this method, since Boxalino tracker send these events by itself.

The choice is an identifier of the test. The variant identifies the selection in the test. The visitor is an identifier of the person that chose the variant. The scope - by default leave this field empty, if the choice you integrate is not coming from Boxalino recommendations, but is generated by another 3rd party software, indicate the name of the scope you use to publish the choice and variants to Boxalino from this software. Extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```
<script type="text/javascript">
    // async
    _bxq.push(['trackChoiceDisplay', 'CHOICEID', 'VARIANTID', 'VISITORID', 'SCOPE']);

    // direct
    _bxq.trackChoiceDisplay('CHOICEID', 'VARIANTID', 'VISITORID', 'SCOPE');
</script>
```

3.19 promise trackLogin(userId, parameters)

Send a login event to the tracker. The userId must be given. There is no need to send this event all the time, but only once per session, when the user logs in.

The userId is an identifier of the user. User identifier is limited to be 50 characters long.

Extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```
<script type="text/javascript">
    // async
    _bxq.push(['trackLogin', 'USERID']);

    // direct
    _bxq.trackLogin('USERID');
</script>
```

3.20 promise trackChoiceClick(choice, parameters)

Send a choice click event to the tracker. Choice id, which is a tracking source, must be provided. This event is used to calculate choice throught rate KPI.

Extra key-value parameters can be appended to the event. Value can be an Array if the parameter is multivalued.

It returns a promise (a function that accept a callback to be called when the tracking is complete) and execute asynchronously.

```
<script type="text/javascript">
    // async
    _bxq.push(['trackChoiceClick', 'CHOICEID']);

    // direct
    _bxq.trackChoiceClick('CHOICEID');
</script>
```