JAVASCRIPT API OVERVIEW

Classification: Public

Author: Ashley Scott (ascott@thunderhead.com)

Version Number: 0.1

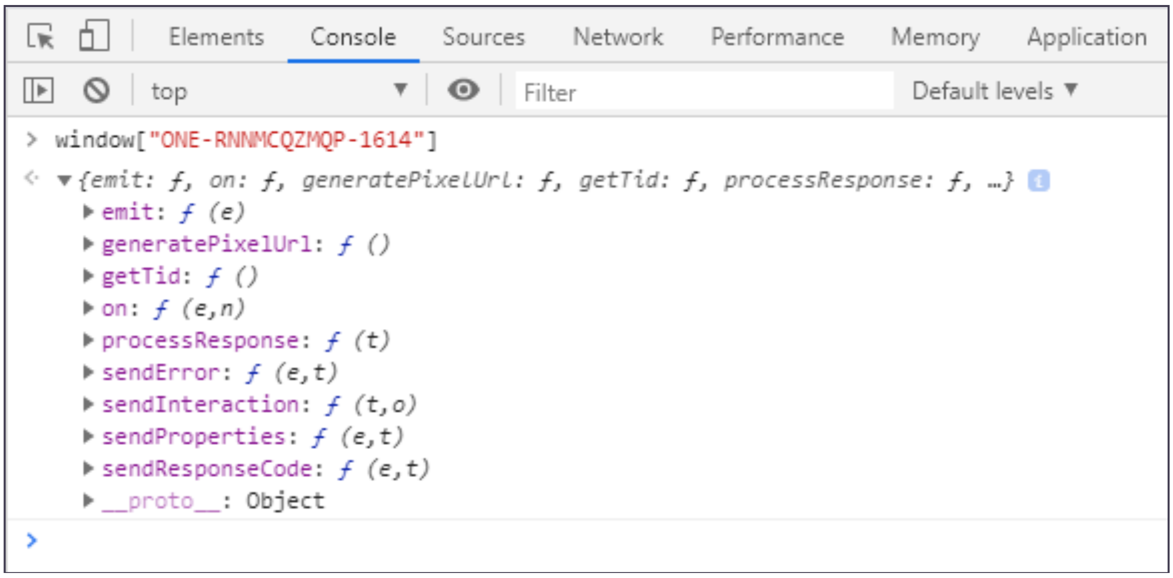
JAVASCRIPT API OVERVIEW

The ONE Tag exposes an object to the window, this will allow the website's code to send interactions to ONE programmatically. If your website has the tag already implemented, you can check if this object is available to you by using the following code example:

```
window["ONE-XXXXXXXX-XXXX"]
```

Replace the example "*ONE-XXXX-XXXX*" with your Site Key. You can find this under "**Configure**" -> "**ONE Tag**".

Open the Developer Tools and navigate to the Console. In here, type the snippet above with your Site Key. You will see the object be displayed with various functions that can be used.



```
> window["ONE-RNNMCQZMQP-1614"]
< ▾{emit: ƒ, on: ƒ, generatePixelUrl: ƒ, getTid: ƒ, processResponse: ƒ, …}
    ▸emit: ƒ (e)
    ▸generatePixelUrl: ƒ ()
    ▸getTid: ƒ ()
    ▸on: ƒ (e,n)
    ▸processResponse: ƒ (t)
    ▸sendError: ƒ (e,t)
    ▸sendInteraction: ƒ (t,o)
    ▸sendProperties: ƒ (e,t)
    ▸sendResponseCode: ƒ (e,t)
    ▸__proto__: Object
>
```

There are many functions, but the important one for tracking events is the "**sendInteraction**" method.

JAVASCRIPT API USAGE

Two parameters are required for the **sendInteraction** method:

1) Interaction URI (string)
2) Properties (object)

Returns:

1) Promise

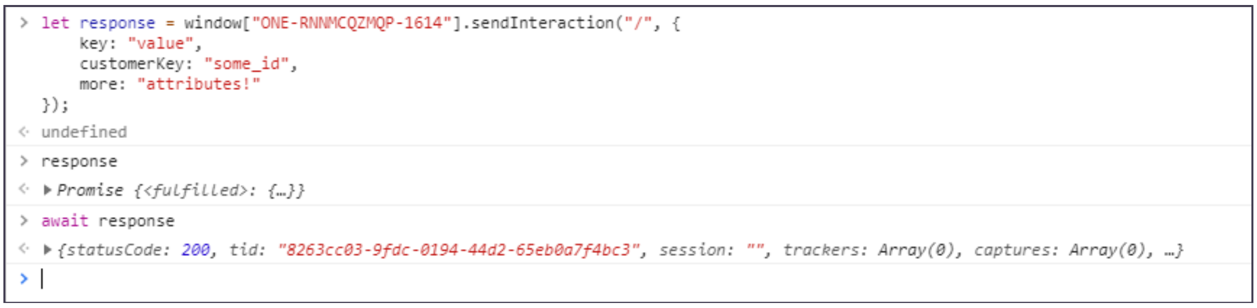
The Interaction URI can be a relative (*/path/to/page*) or absolute (*https://example.com/path/to/page*) URL. The relative path should be used in most cases.

The Properties object contains the customer/marketing attribute values that you want to store against the customer's profile. These essentially act like query string parameters when ONE receives them. (Any query string parameters in the URL are also sent to ONE). An example of what this looks like:

```
[...]
// Vue component has been created, or similar render
//  event for other frameworks like Angular or React
created() {
    // Send an interaction when the view for this page.
    window["ONE-XXXXXXXX-XXXX"].sendInteraction("/path/to/page", {
        key: "value",
        customerKey: "some_id",
        more: "attributes!"
    })
}
[...]
```

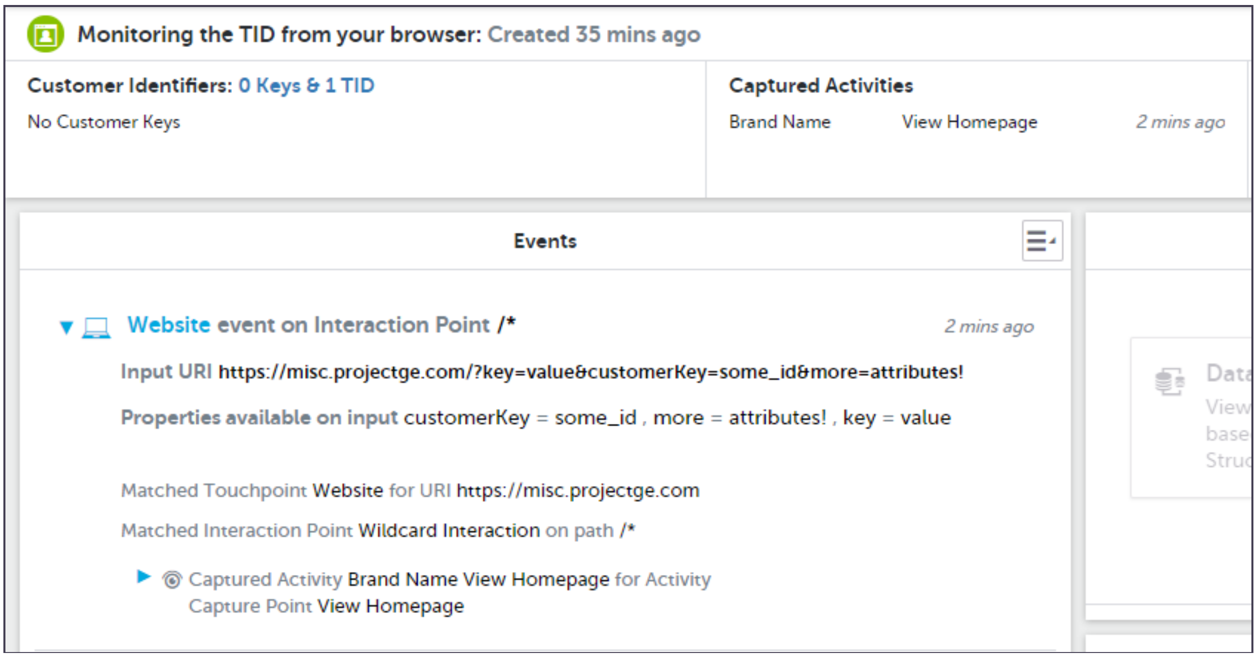
The returned promise can be awaited, or you can use "*.then(response => {…})*" to handle ONE's response. (More info below!)

Running the code example in your browser:



```
> let response = window["ONE-RNNMCQZMQP-1614"].sendInteraction("/", {
      key: "value",
      customerKey: "some_id",
      more: "attributes!"
  });
< undefined
> response
< ▸Promise {<fulfilled>: {…}}
> await response
< ▸{statusCode: 200, tid: "8263cc03-9fdc-0194-44d2-65eb0a7f4bc3", session: "", trackers: Array(0), captures: Array(0), …}
> |
```

Login to ONE and use “Test & Publish” -> “Monitor” to verify that your event has been tracked!



Monitoring the TID from your browser: Created 35 mins ago

Customer Identifiers: 0 Keys & 1 TID
No Customer Keys

Captured Activities
Brand Name   View Homepage   2 mins ago

Events

▾ Website event on Interaction Point /*   2 mins ago

Input URI https://misc.projectge.com/?key=value&customerKey=some_id&more=attributes!

Properties available on input customerKey = some_id , more = attributes! , key = value

Matched Touchpoint Website for URI https://misc.projectge.com

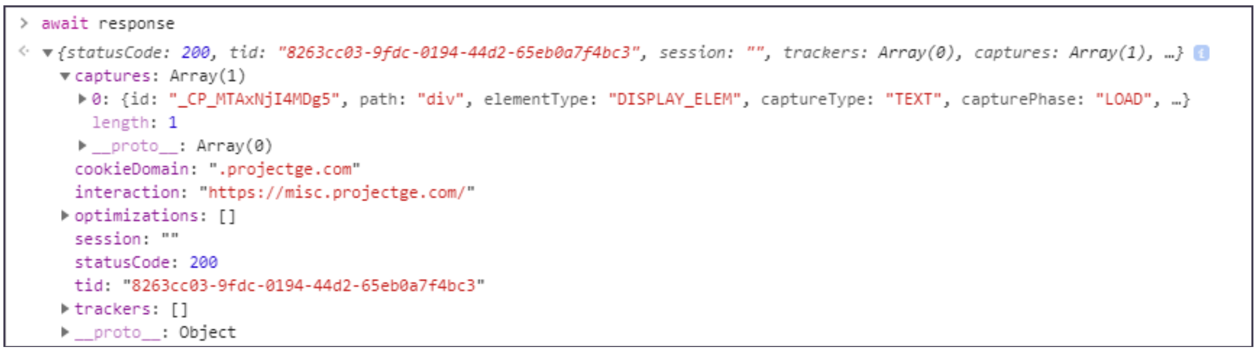
Matched Interaction Point Wildcard Interaction on path /*

▸ Captured Activity Brand Name View Homepage for Activity Capture Point View Homepage

Notice the “*Properties available on input*”, this matches the data sent in the “**sendInteraction**” properties payload. If your interaction point has Attribute Capture Points for these parameters, they will be stored in the profile!

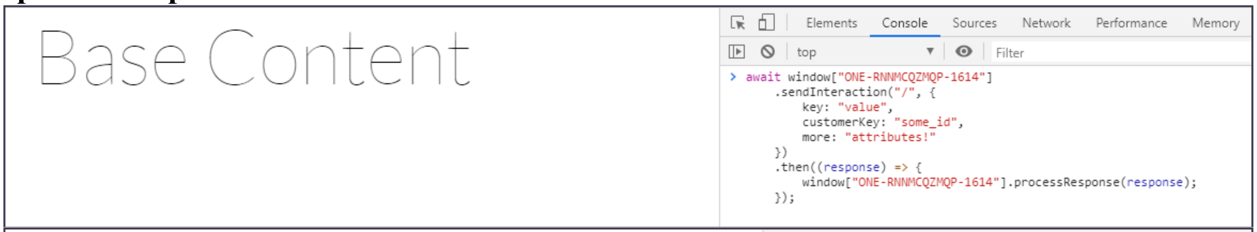
JAVASCRIPT API RESPONSE

The “**sendInteraction**” method returns a response that can be used by another function called “**processResponse**” which will apply the click tracker, input capture and optimisation points. From the previous code example, observe the response given by ONE:

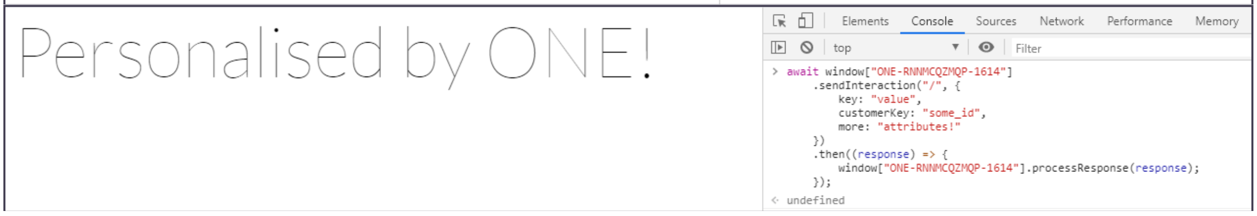


```
> await response
< ▼{statusCode: 200, tid: "8263cc03-9fdc-0194-44d2-65eb0a7f4bc3", session: "", trackers: Array(0), captures: Array(1), …}
   ▼captures: Array(1)
     ▶0: {id: "_CP_MTAxNjI4MDg5", path: "div", elementType: "DISPLAY_ELEM", captureType: "TEXT", capturePhase: "LOAD", …}
      length: 1
     ▶__proto__: Array(0)
    cookieDomain: ".projectge.com"
    interaction: "https://misc.projectge.com/"
   ▶optimizations: []
    session: ""
    statusCode: 200
    tid: "8263cc03-9fdc-0194-44d2-65eb0a7f4bc3"
   ▶trackers: []
   ▶__proto__: Object
```

These capture/optimisation points can be applied manually, but it is recommended to use the “**processResponse**” in most cases:



```
> await window["ONE-RNNMCQZMQP-1614"]
      .sendInteraction("/", {
          key: "value",
          customerKey: "some_id",
          more: "attributes!"
      })
      .then((response) => {
          window["ONE-RNNMCQZMQP-1614"].processResponse(response);
      });
```



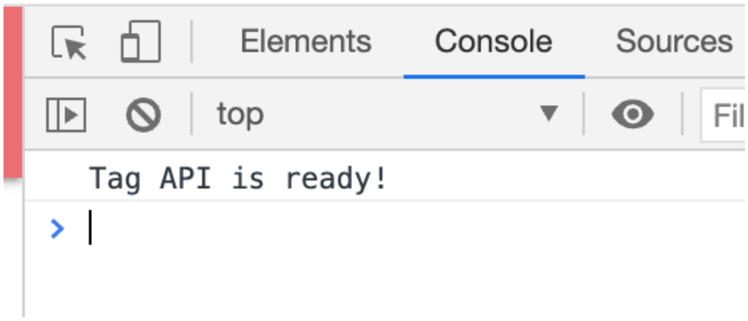
```
> await window["ONE-RNNMCQZMQP-1614"]
      .sendInteraction("/", {
          key: "value",
          customerKey: "some_id",
          more: "attributes!"
      })
      .then((response) => {
          window["ONE-RNNMCQZMQP-1614"].processResponse(response);
      });
< undefined
```

API READY EVENT LISTENER

When the tag is injected into the page, and the API becomes available, an event is emitted. Similarly to how the tag APIs are exposed via **window[“SITE_KEY”]**, the type of event to listen for is **“SITE_KEY”**. To set this up, you can use the following code to subscribe to the event:

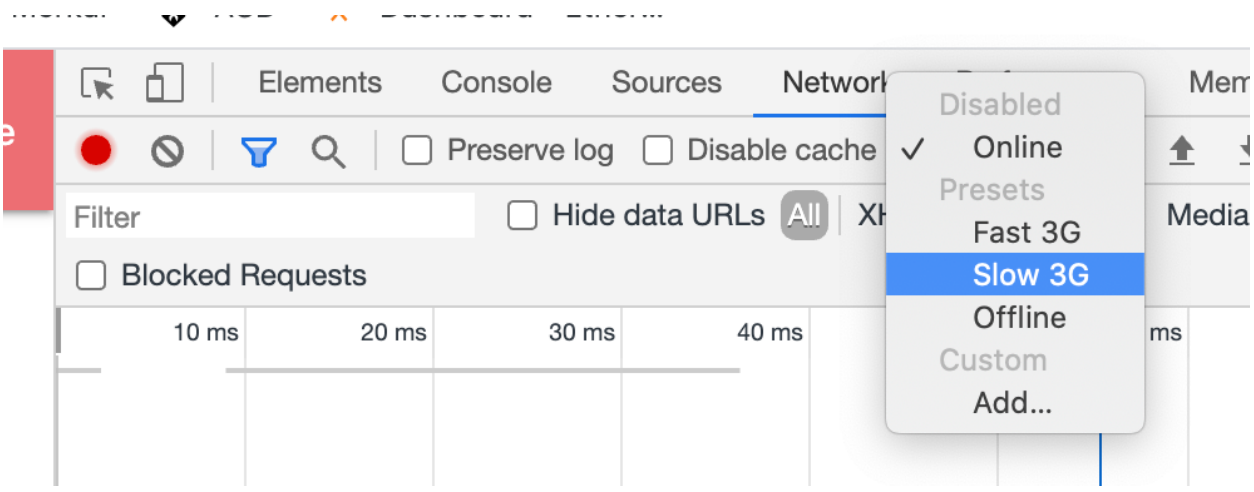
```
document.addEventListener("ONE-KB7YLIQSRT-8592",function () {
    console.log("Tag API is ready!");
});
```

Using the above example, when the website loads and the tag api becomes available, a message will be displayed:



```
Tag API is ready!
```

Try changing the network speed in the Dev Tools under “Network”. From the dropdown list next to “Disable cahce”, set the speed to “Slow 3G”.
Switch back to the console and reload the page.



Notice how the tag takes longer to initialise as the message doesn’t appear as quickly.
This will protect your events from race conditions, ensuring that the tag is always available before sending an interaction.

SUMMARY

… and that’s it! This JavaScript object can be called when needed, but ideally on the render of the page.
If you only need to send some attributes without triggering an interaction, the “**sendProperties**” method can be used with the exact same method signature, excluding the need to “**processResponse**”.
If there are any questions, don’t hesitate to reach out to us via email. Or use the customer support portal on the website: https://thunderhead.jira.com/servicedesk/customer/portal/1

TO LEARN MORE ABOUT THUNDERHEAD VISIT:

@thunderheadONE

Linkedin.com/company.thunderhead---com

conversations@thunderhead.com

THUNDER HEAD

INTENT-DRIVEN
CUSTOMER JOURNEYS

THUNDERHEAD.COM