Decibel

Implementation Guide v5.1

Contents

Contact Information

For further assistance during implementation, please contact Decibel using one of the following methods.

Support Tickets

To create a support ticket, log in to Decibel and select the *Help > Support* menu option, and a member of our Technical Support team will assist you.

E-mail

Please e-mail your Customer Success Manager directly, or contact the Technical Support team at support@decibelinsight.com

Telephone

Please phone your Customer Success Manager directly, or call your local office on:

+1 917 979 5770
+44 20 3700 4700

Implementation Checklist

To assist with the implementation process, the following checklist can be completed to ensure that all required steps are undertaken.

Task	Complete?	Reference
Step 1 – Property Configuration		
Configure Properties		Page 5
Configure ePrivacy Compliance Options		Page 5
Step 2 – Tag Installation		
Deploy Tag		Page 7
Validate Tag Installation		Page 7
Exclude Internal IP Addresses		Page 7
Step 3 – User Account Setup		
Configure Users		Page 8
Step 4 – Goal Tracking		
Configure Goals		Page 9
Trigger Goals		Page 10
Step 5 – Form Tracking		
Select Collection Mode		Page 12
Tag Forms		Page 12
Configure Validation Error Tracking		Page 13
Configure Data Masking		Page 15
Step 6 – Error Tracking		
Configure HTTP Server Error Tracking		Page 16
Configure Application Error Tracking		Page 17
Step 7 – Custom Dimensions		
Configure Custom Dimensions		Page 18
Send Custom Dimension Data		Page 19
Step 8 – Sampling Configuration		
Configure Sample Rates		Page 20
Step 9 – Additional Configurations		
Configure On-page Data Masking		Page 23
Configure Regex Paths		Page 24

Step 1 – Property Configuration

Each property that you wish to track using Decibel must first be configured within the Decibel Portal. A property may be a desktop, mobile or responsive website, or a hybrid mobile app.

1. Log in to Decibel and select the *Settings* > *Properties* menu option.
2. Click on *Add Property*, select the property type and enter the website domain or name of the mobile app. When configuring a website, it is important to use the domain that shows in the browser navigation bar when browsing the website.
3. If the Property is a website, enter any additional sub-domains used, for example, your website may use “www.mywebsite.com” and “secure.mywebsite.com”.
4. Configure any additional settings for the property. In most cases, these can be left as the default values. The later sections of this guide will explain these options in more detail.
5. Click *Save Property* to validate and save the property configuration.

If the *Add Property* option does not appear, you may have reached the limit for properties on your subscription. Please contact your Customer Success Manager to arrange additional properties.

Multiple Property Configuration

A single Decibel account may be used to track multiple websites and apps. Configure each additional property in the *Settings* > *Properties* screen and use the individually generated tags for each property, as described in the next step.

ePrivacy Compliance Configuration

By default, Decibel uses cookies and tracks IP addresses to detect return visitors to a website. It is possible to configure Decibel to not use cookies and to drop or anonymize IP addresses where this is a requirement of specific regional legislation.

1. Log in to Decibel and select the *Settings* > *Properties* menu option.
2. Edit the required Property and select the *Privacy* tab.
3. Untick the *Set Cookies* checkbox if you do not wish to set cookies on visitor’s browsers. Note that this will remove the ability for Decibel to track return visits to the website.
4. Select the desired option for *IP Address Handling* to either drop, anonymize or store the IP address for visitors to the website.
5. Click *Save Property* to save the updated configuration.

Step 2 – Tag Installation

The Decibel tag should be added to every page or screen on the website or app; either through a tag manager or directly within your website template.

1. Log in to Decibel and select the *Settings* > *Tracking Code* menu option.

2. Copy the tag and insert within your HTML <head> block on each page, immediately before any other <script> tags, for example:

```
<html>
  <head>

  <!-- title and meta -->

  <script type="text/javascript">
  // <![CDATA[
  (function(d,e,c,i,b,el,it) {
    d._da_=d._da_||[];_da_.oldErr=d.onerror;_da_.err=[];
    d.onerror=function(){_da_.err.push(arguments);
    _da_.oldErr&&_da_.oldErr.apply(d,Array.prototype.slice.call(arguments));};
    d.DecibelInsight=b;d[b]=d[b]||function(){(d[b].q=d[b].q||[]).push(arguments);};
    el=e.createElement(c),it=e.getElementsByTagName(c)[0];
    el.async=1;el.src=i;it.parentNode.insertBefore(el,it);
  })(window,document,'script','https://cdn.decibelinsight.net/i/XXX/YYY/di.js','decibelInsight');
  // ]]>
  </script>

  <!-- additional scripts -->
  </head>
```

In the above example, **XXX** indicates your account ID and **YYY** indicates the property ID. These placeholders will contain numeric values when the tag is copied from the *Settings* > *Tracking Code* menu option.

It is important that the Decibel tag is placed before other script tags, as Decibel will then be able to record and understand more of the visitor journey. The script loads asynchronously, so it won't slow down your website.

Tag Manager Implementation

Decibel may be installed using a tag manager rather than directly deploying the tag to the website. Default templates exist in all major tag management platforms, or alternatively the above tag may be deployed directly within a tag container.

For further assistance with deployment through a tag manager, please contact Decibel's technical support team or your tag management vendor.

Website Implementation Validation

Once implemented on your website, Decibel will automatically run tests to verify that your implementation is correct. You can check this manually by going to *Settings > Tracking Code* and clicking *Check my tracking code*.

Decibel will only track a website if the domain has been registered. If you are testing the tag on a staging website, please make sure you register the domain first in the *Settings > Properties* screen.

Blocking IP Addresses

Decibel can be configured to exclude visitors that you do not want to track based on their IP address. This is often used to exclude traffic from internal networks and known crawlers or robots.

1. Log in to Decibel and select the *Settings > Blocked IPs* menu option.
2. Click on *Add Blocked IP*, entering the IP address or a regular expression matching one or more IP addresses. You may also enter a description to explain the purpose of the Blocked IP to other Decibel users.
3. Click *Save Blocked IP* to create the Blocked IP address configuration.

There is a limit of 250 Blocked IPs per account. Decibel may also automatically create Blocked IPs if unusual traffic patterns are detected while processing data.

Specific Implementation Scenarios

Single Page Application Tracking

Decibel can be implemented on Single Page Applications (SPAs) using technologies such as AngularJS or Backbone.js by deploying the same tag. If the application is designed using the HTML 5 History API, virtual page views will be automatically detected and tracked by Decibel.

If the application does not change the URL to denote virtual page views, it will be necessary to use the `trackPageView` JavaScript API end-point. See the *JavaScript API* section below for further details.

Step 3 – User Account Setup

Additional user accounts can be configured to allow other team members access to configure properties and/or view reports on configured properties.

1. Log in to Decibel and select the *Settings > Users and Access* menu option.
2. Click on *Add User* and enter the new user's first name, last name and email address.
3. Select the *Roles* for the new user – this defines the features, reports and properties that the user will have access to. To assign permissions directly to the user, select the *Custom* role, or choose the *Administrator* role if the user requires full access to the account.
4. Click *Save User* to save the user account configuration. An email will be sent to the user containing a link which enables them to configure their password and gain access to the Decibel Portal.

If the *Add User* option does not appear, you may have reached the limit for user accounts on your subscription. Please contact your Customer Success Manager to arrange additional user accounts.

Step 4 – Goal Tracking

Goals are events that occur within a visitor's journey that are of interest when analyzing those journeys. Goals may optionally have an attached value (for example, the value of the products or services the visitor purchased).

An example of a goal may be downloading a whitepaper, viewing a video, completing a contact form, registering for an account, making a purchase, or interacting with a specific piece of content.

Configuring Goals

Goals must be configured within your Decibel account.

1. Log in to Decibel and select the *Settings > Goals* menu option.

2. Click on *Add Goal*, entering a *Name* and an optional *Description* to explain the purpose of the Goal to other Decibel users.

3. Select the *Status* to be assigned to the visitor once this goal has been achieved. For example, on an e-commerce website, the “Add to Basket” goal may have a status of “Engaged”, while the “Checkout” goal would have a status of “Converted”.

4. Select the *Integration method* for triggering the Goal:

    - *Decibel JavaScript API Call* – to trigger the Goal using an on-page call to the Decibel JavaScript API, as described below.
    - *JavaScript On-click Event* – this option allows the Goal to be automatically triggered whenever a specific element is clicked on. If selected, in the Selector text box, enter a CSS selector that identifies the HTML element(s) that will trigger the Goal when clicked on (for example, “#checkout” or “div.product > .addToBasket”).

5. Click *Save Goal* to create the goal configuration.

Triggering Goals with JavaScript

In some cases, a goal may need to be tracked based on a user's interaction on a web page, for example clicking on a Checkout button in a shopping cart. In these scenarios, the Decibel JavaScript API is the most appropriate method for tracking a goal.

1. Log in to Decibel and select the *Settings* > *Goals* menu option.
2. Click the *Show integration code* action next to the appropriate Goal.
3. Copy the displayed integration code and implement it using a Tag Manager or directly within the website code. The following example shows a goal being triggered when an element on the page is clicked:

```
    </div>
    <!-- end content -->

    <script type="text/javascript">
      $('#checkoutButton').click(function() {
        decibelInsight('sendGoal', 'Checkout', dataLayer.basketValue);
      });
    </script>
  </body>
</html>
```

The `sendGoal` end-point triggers the goal within Decibel. The second parameter is the goal name, while the third parameter provides the monetary value of the goal and may be omitted. In this example, the third parameter refers to a JavaScript variable that has been defined in the website's data layer, containing the monetary value of the transaction.

See the *JavaScript API* section below for further details.

Triggering Goals with Rules

Using Rules, a Goal can be triggered when a certain condition is met, such as viewing a specific page, spending a certain amount of time on the website, or making a return visit to the website.

1. Log in to Decibel and select the *Settings* > *Rules* menu option.
2. Click on *Add Rule* and select the appropriate condition for the Goal to be triggered.
3. In the *Action* section, select *Trigger Goal* and select the appropriate Goal from the drop-down.
4. Click *Save Rule* to create the rule configuration. Note that rules are not retroactive, so this will only apply to future sessions.

Step 5 – Form Analytics

Decibel's Form Analytics monitors and reports on your forms, showing how effective each form is, and breaking down the performance of each field within it.

Understanding Dynamic and Tagged Tracking

Decibel can dynamically collect data for all forms on a website; however, if forms aren't well named it may be necessary to tag them so that the collected data is meaningful.

The following example shows a well-named form that could be tracked dynamically by Decibel. It has an understandable `id` attribute for the form itself, as well as HTML labels and understandable `name` attributes on each of the fields:

```
<form action="submit.php" method="POST" id="contact">
  <label>First name: <input type="text" name="first_name"></label>
  <label>Last name: <input type="text" name="last_name"></label>
  <input type="submit" value="Submit">
</form>
```

The next example shows a form that is not well named and would need to be tagged to be tracked in a meaningful way. This form has meaningless `name` attributes on the fields, there are no labels to indicate the field names, and the form element itself is missing an `id` attribute:

```
<form action="submit.php" method="POST">
  First name: <input type="text" name="field156"><br />
  Last name:  <input type="text" name="field135"><br />
  <input type="submit" value="Submit">
</form>
```

Other examples of forms that may need to be tagged include:

- Forms that have different fields or different numbers of fields depending upon the visitor's journey (for example, a checkout form that contains fields to customize specific products in the basket).
- Forms that use randomized or session-based `id` or `name` attributes on the form or field elements.
- Forms that are presented on multiple pages throughout the website with the same fields, however using different `id` or `name` attributes on the form or field elements on different pages.

Selecting the Collection Mode

After reviewing the forms on the website and deciding upon a collection mode, this must be selected to enable Form Analytics.

1. Log in to Decibel and select the *Forms* menu option.

2. Click the *Choose Custom Setup* or *Choose Automatic* Setup option depending on whether you would like to use tagged or dynamic collection respectively.

If you are uncertain about which collection mode to use, we recommend starting with dynamic collection. If it later becomes obvious that tagged collection is required, please contact Technical Support who can clear existing data and switch the collection mode.

Tagging Forms

If tagged collection has been selected, the appropriate data attributes will need to be added to each of the forms on the website that require tracking. These attributes provide consistent and understandable names to each of the forms and fields being tracked. Below is an example of a form with tags in place:

```
<form action="submit.php" method="POST" data-di-form-track data-di-form-id="contact">
  <div>
    <span>First name:</span>
    <input type="text" name="field156" data-di-field-id="firstname">
  </div>
  <div>
    <span>Last name:</span>
    <input type="text" name="field135" data-di-field-id="lastname">
  </div>
  <input type="submit" value="Submit">
</form>
```

The following table describes the available form tags:

Tag Name	Description
`data-di-form-track`	If this attribute is present, the form will be tracked.
`data-di-form-id`	The value of this attribute will be used as the form's name within Decibel. If this is not present and `data-di-form-track` (above) is present, then we will use the existing mechanism of determining the form ID. This tag can still be used if collection is dynamic.
`data-di-field-ignore`	If this attribute is present, the field will not be tracked. Please note: Button, Submit, Reset, Image and Hidden fields are always ignored.
`data-di-field-id`	The value of this will be the field ID. If not present, Decibel will use the input's `id` attribute, or `name` attribute if no `id` is present.

Configuring Form Validation Error Tracking

Decibel can report on the forms and fields on which visitors receive validation errors. As there is no native way to mark-up validation errors in HTML, this requires some custom configuration, achieved using tagging or by writing a custom JavaScript function within property configuration.

Decibel will attempt to locate the error message for each field in the form in the following order:

1. The field will be passed to the Field Error Callback function.
2. If the Field Error Callback returns false, the `data-di-field-error` and `data-di-field-error-for` attributes will be checked as described below.

Field Error Callback Function

The Field Error Callback is a custom JavaScript function that can be used to identify form errors.

1. Log in to Decibel and select the *Settings > Properties* menu option.
2. Edit the appropriate property and select the *Form* tab.
3. In the *Field Error Callback* text box, enter the body of a JavaScript function that will return:
   - `false` if there is no validation error present for `field`
   - `true` if there is a validation error present for `field`, but no specific error message, or
   - the error message text if there is a specific error message available

   See the example below for more details.
4. Click *Update Property* to save the updated configuration.

Below is an example form containing a validation error message, and subsequently an example function that can be used to configure the tracking of these error messages:

```
<form action="submit.php" method="POST" id="contact">
  <label>First name: <input type="text" name="first_name">
    <span class="error">First name must be provided</span>
  </label>
  <label>Last name: <input type="text" name="last_name"></label>
  <input type="submit" value="Submit">
</form>
```

```
function (field) {
    // Locate the error element by finding the first sibling with a "error" class
    var error = field.parentNode ? field.parentNode.querySelector('.error') : false;

    // If no error element exists, end the function by returning false
    if(!error) return false;

    // If an error exists and has text, trim the text and return the message
    return error.innerText ? error.innerText.trim() : false;
  }
```

Form Error Tagging

If tagged collection has been selected, additional tags can be used to indicate when fields are in error state, and to point out the relevant error message for each field of the form. Below is an example of a form with tags in place:

```
<form action="submit.php" method="POST" data-di-form-track data-di-form-id="contact">
  <div>
    <span>First name:</span>
    <input type="text" name="field156" data-di-field-error data-di-field-id="firstname">
    <span class=”error” data-di-field-error-for="firstname">Please enter your first
  name.</span>
  </div>
  <div>
    <span>Last name:</span>
    <input type="text" name="field135" data-di-field-id="lastname">
  </div>
  <input type="submit" value="Submit">
</form>
```

The following table describes the available validation error tags:

Tag Name	Description
`data-di-field-error`	If this attribute is present, it indicates that the field is in error state. A string value will be used as the error message, or if no value is present the `data-di-field-error-for` tag will indicate the error message.
`data-di-field-error-for`	Indicates that text within the tagged element is an error message for the field with the specified ID.

Please contact your Customer Success Manager if you would like assistance with the configuration of form validation error message tracking for your website.

Configuring Data Masking

By default, all information entered into forms by visitors is masked on the visitor's browser, and never sent to Decibel. In cases where fields do not contain Personally Identifiable Information (PII), it may be desirable to record this information to assist with behavioral analysis.

1. Log in to Decibel and select the *Settings* > *Properties* menu option.
2. Edit the appropriate property and select the *Privacy* tab.
3. In the *Unmasking Field Selector* text box, enter a CSS selector defining the fields that should be unmasked. For example, the CSS selector “[name="company"],.noPII” would unmask any fields with a `name` attribute matching “company” or a `class` matching “noPII”. The *Unmasking Field Selector* applies to all forms being tracked on the property.

As the Decibel masking algorithm is not reversible, this change will not be retroactive and will only apply to session recordings going forward.

Step 6 – Error Tracking

Decibel provides reports to track multiple types of errors that may be experienced by visitors on a website:

- **JavaScript Errors** – Any JavaScript errors triggered during a visitor's session will be tracked by default and require no configuration.
- **Form Validation Error Messages** –Validation errors on form fields can be tracked and must be configured as described in the previous step.
- **HTTP Server Errors** – Low level errors returned by the application web server, such as *404 Page Not Found* and *500 Internal Server Error* can be tracked by adding a tag to the server error pages, explained below.
- **Application Errors** – Any other error messages exposed by the application, for example, a message shown after checkout completion to explain that a selected product is no longer in stock. These can be tracked through the Decibel JavaScript API as explained below.

Configuring HTTP Server Error Tracking

To track HTTP server errors, the Decibel `sendHTTPError` JavaScript API end-point must be called on server error message pages, in addition to firing the main Decibel tag for the property being tracked:

```
<html>
  <head>
  <title>503 Internal Server Error</title>

  <script type="text/javascript">
  // <![CDATA[
  (function(d,e,c,i,b,el,it) {
    d._da_=d._da_||[];_da_.oldErr=d.onerror;_da_.err=[];
    d.onerror=function(){_da_.err.push(arguments);
    _da_.oldErr&&_da_.oldErr.apply(d,Array.prototype.slice.call(arguments));};
    d.DecibelInsight=b;d[b]=d[b]||function(){(d[b].q=d[b].q||[]).push(arguments);};
    el=e.createElement(c),it=e.getElementsByTagName(c)[0];
    el.async=1;el.src=i;it.parentNode.insertBefore(el,it);
  })(window,document,'script','https://cdn.decibelinsight.net/i/XXX/YYY/di.js','decibelInsight');
  decibelInsight('sendHTTPError', 503);
  // ]]>
  </script>

  </head>
  <body>
    <h1>503 Internal Server Error</h1>
    ...
  </body>
</html>
```

Configuring Application Error Messages

Application Error messages are collected using the Decibel `sendApplicationError` JavaScript API end-point. This can be called directly on the page when a message is shown, using one of the following methods.

To provide a single error message to Decibel, simply pass the error message text (this can be hard-coded or retrieved from a data layer JavaScript variable), for example:

```
    </div>
    <!-- end content -->

    <script type="text/javascript">
      decibelInsight('sendApplicationError', dataLayer.errorMessage);
    </script>
  </body>
</html>
```

Alternatively, a CSS selector can be provided which matches all HTML elements on the page that may contain an Application Error message. In this case, zero or more error messages that match the provided CSS selector will be tracked by Decibel.

```
    </div>
    <!-- end content -->

    <script type="text/javascript">
      decibelInsight('sendApplicationError', '.error', true);
    </script>
  </body>
</html>
```

See the *JavaScript API* section below for further details.

Step 7 – Custom Dimensions

Custom Dimensions are used to store additional information for a visitor, session or page view within Decibel. Information stored in Custom Dimensions can be demographic (for example gender or age range), product related (for example product codes or order IDs) or customer specific (such as a customer reward program number or tier).

Configuring Custom Dimensions

Custom Dimensions must be configured within your Decibel account.

1. Log in to Decibel and select the *Settings > Custom Dimensions* menu option.

2. Click on *Add Custom Dimension*, entering a *Name* and an optional *Description* to explain the purpose of the Custom Dimension to other Decibel users.

3. Select the appropriate *Type* for the data you wish to store in the Custom Dimension:

    - *String* – for textual data
    - *Boolean* – for yes/no or true/false values
    - *Number* – for numerical values
    - *Choice* – for a pre-defined list of options. If *Choice* is selected, you must also enter a comma-separated list of choices (for example, “Gold,Silver,Bronze”)

4. Select the appropriate *Scope* for the Custom Dimension:

    - *Page view* – for data that applies to the current page view only, for example the Product ID of the product shown on the current page.
    - *Session* – for data that applies to the current session, for example the Order ID of a completed transaction.
    - *Visitor* – for data that applies to the current visitor across all their sessions, for example their reward program tier.

5. Select the *Integration method* for Custom Dimension data retrieval:

    - *Decibel JavaScript API Call* – to push the Custom Dimension data using an on-page call to the Decibel JavaScript API, as described below.
    - *Load from JavaScript Variable* – for data that is exposed in a JavaScript variable (for example the data layer) on the page. If selected, enter the name of the JavaScript variable containing the data.
    - *Retrieve from URL Parameter* – if the value is present in a query string parameter in the page URL. If selected, enter the name of the URL parameter containing the data.

6. Click *Save Custom Dimension* to create the Custom Dimension configuration.

Sending Custom Dimension Data

If the selected *Integration* method for a Custom Dimension is “Decibel JavaScript API Call”, the correct JavaScript code must be added to the website.

1. Log in to Decibel and select the *Settings* > *Custom Dimensions* menu option.
2. Click the *Show integration code* action next to the appropriate Custom Dimension.
3. Copy the displayed integration code and implement it using a tag manager, or directly within the website code. The following example shows a Custom Dimension value being sent when an element on the page is clicked:

```
    </div>
    <!-- end content -->

    <script type="text/javascript">
      $('#addToBasket').click(function() {
        decibelInsight('sendCustomDimension', 'Basket Value', $('#total').text());
      });
    </script>
  </body>
</html>
```

The `sendCustomDimension` end-point sends Custom Dimension data to Decibel. The second parameter is the Custom Dimension name and the third parameter contains the value of the custom dimension to be set.

See the *JavaScript API* section below for further details.

Step 8 - Sampling Configuration

If enabled on your subscription, it is possible to configure the sample rates controlling how much data is collected, processed and stored by Decibel.

Collection Sample Rate and Retention Sample Rate

The *Collection Sample Rate* controls the number of visitor sessions that are selected for tracking by Decibel. A Collection Sample Rate of 50%, for example, would track every second visitor to the website, while a sample rate of 0.1% would track one out of every 1000 visitors.

It is also possible to configure whether return visitors should automatically be included in the Collection Sample. If enabled, any visitor that is initially included in the sample will continue to be included for each of their subsequent visits to the website. When disabled, return visits will be included only if that session is randomly selected to be included in the sample at the time of the visit.

The *Retention Sample Rate* controls the number of collected visitor sessions that are stored for use in Decibel reports and as session replays. A Retention Sample Rate of 25% would result in one in every four collected sessions to be stored for later use, while the other three sessions would be discarded on completion.

The Retention Sample is most useful on very high traffic websites, when combined with the `setRetention` JavaScript API end-point, which is explained in further detail below.

When both enabled, the Collection Sample Rate and Retention Sample Rate combined will determine the total number of sessions available for reporting purposes within Decibel.

The following table provides some example scenarios showing the actual number of website visits compared to the number of sessions that will be processed and retained by Decibel:

Actual Sessions	Collection Sample Rate	Retention Sample Rate	Retained Sessions
2.5m	100%	100%	2.5m
22m	25%	100%	5.5m
70m	20%	80%	11.2m
240m	15%	50%	18m

Collection and Retention Limits

Your subscription will include a limit on the number of data credits that may be collected and retained monthly. If this limit is less than the total volume of traffic received by your website, it is important to configure the sample rates to ensure that this allowance is not completely used before the end of the month.

Where multiple properties are configured in an account, it is also possible to set upper limits on the number of data credits that will be collected and retained by an individual property, to ensure that it doesn't consume the total allowance for the account.

Configuring Sample Rates and Retention Limits

Sample rates can be configured independently for each tracked property. It is also possible to adjust sample rates throughout the subscription to control usage of the data credit allowances for the account.

1. Log in to Decibel and select the *Settings* > *Properties* menu option.
2. Edit the appropriate property and click the *Sampling* tab.
3. Enter the desired *Collection Sample Rate* and *Retention Sample Rate* as percentage values between "0.00001" and "100"
4. Set the *Automatically Include Return Visitors* checkbox to determine whether return visits will always be included in the sample.
5. Enter the desired *Collection Limit* and *Retention Limit* or tick the corresponding *Unlimited* checkbox if no limits are required. These limits must be less than or equal to the total collection and retention limits for the account.
6. Click *Save Property* to save the updated sampling configuration.

If any of the above options do not appear, they may not be enabled on your subscription. Please contact your Customer Success Manager to discuss available sampling options.

Forcing Collection and Retention

Using the Decibel JavaScript API, it is possible to force specific sessions to be collected or retained, regardless of whether they were automatically selected to be included in the sample.

The following example shows a session being marked for retention following a checkout button being clicked on the website:

```
    </div>
    <!-- end content -->

    <script type="text/javascript">
      $('#checkoutButton').click(function() {
        decibelInsight('setRetention', true);
      });
    </script>
  </body>
</html>
```

In this example, the `setRetention` JavaScript API end-point is used, which will result in the entire session for this visitor (including any page views prior to the current page view) being processed and retained by Decibel. Alternatively, the `setCollection` JavaScript API end-point can be used to record page views from this point forward in the journey.

See the *JavaScript API* section below for further details.

Step 9 – Additional Configuration

The following configurations may be required depending on the nature of the property being tracked. If unsure, please contact your Customer Success Manager who can assist further.

Configuring On-page Data Masking

Where Personally Identifiable Information (PII) is shown to visitors within the web page HTML (for example login areas or confirmation pages), it is possible to configure Decibel to mask this information.

1. Log in to Decibel and select the *Settings* > *Properties* menu option.
2. Edit the appropriate property and select the *Privacy* tab.
3. Select the required automatic masking patterns by ticking the following checkboxes:
    - *Mask Credit Card Numbers* (this is mandatory and cannot be disabled)
    - *Mask Email Addresses*
    - *Mask Social Security Numbers*
4. In the *Masking Regex* text box, enter any custom regular expressions to mask consistently formatted personal information on the page. For example, to mask Contract IDs like "D77-GH2-5D8", enter the regular expression "[A-Z0-9]{3}-[A-Z0-9]{3}-[A-Z0-9]{3}"
5. In the *Masking Query Selector* text box, enter a CSS selector defining the HTML elements that should be masked. For example, the CSS selector ".pii,#email" would mask any text content or images contained within elements with an `id` attribute matching "email" or a `class` matching "pii"
6. Click the *Save Property* button to save the updated property configuration.

You may use any combination of the above mechanisms for masking, according to the needs of the property being tracked.

As the Decibel masking algorithm is not reversible, this change will not be retroactive and will only apply to session recordings going forward.

Configuring Regex Paths

Regex (or Regular Expression) Paths are used to describe patterns where multiple URLs may identify a single page on the website. For example, on a billing system, the customer's Contract ID may be used within the URL structure, as shown in the following sample URLs:

- www.mywebsite.com/Billing/374-3D3-AK4/
- www.mywebsite.com/Billing/F92-K52-V88/
- www.mywebsite.com/Billing/98D-NMK-3S2/

A regular expression may be written to identify and normalize these URLs, so that aggregated data for these pages can be shown in heatmaps and other Decibel reports. In this example, the following regular expressions could be used:

Regular Expression	Replacement
^Billing/[A-Z0-9]{3}-[A-Z0-9]{3}-[A-Z0-9]{3}/$	Billing/
^(.+)/[A-Z0-9]{3}-[A-Z0-9]{3}-[A-Z0-9]{3}/$	$1/

The first regular expression is more precise, only matching URLs that start with "Billing", while the second regular expression will match any URL that ends with a Contract ID and retain the start of the URL. In both examples, all three URLs above will be converted to "www.mywebsite.com/Billing/".

Regex Paths must be configured in the Decibel Portal, after which the rule will apply to all subsequently recorded page views.

1. Log in to Decibel and select the *Settings* > *Regex Paths* menu option.
2. Click on *Add Regex Path*, entering the regular expression and replacement.
3. Click *Save Regex Path* to create the Regex Path configuration.

Appendix A: JavaScript API

This section describes the functionality available within the Decibel JavaScript API. These end-points can be called from within the pages of your website to further enhance or customize the functionality of Decibel.

Events

The following events can be used to trigger website functionality and other data integrations based on Decibel behaviors.

`decibelInsight('ready', callback);`

In some cases, JavaScript code needs to be executed after Decibel has been fully loaded. This event allows custom code to be executed once this condition has occurred. You can bind one or more callbacks which will be executed in the same order they have been bound.

Arguments

- `callback` (function): A callback containing any valid JavaScript code.

Example

```
// Log a message in the console once Decibel is fully loaded.
decibelInsight('ready', function () {
  console.log('Decibel loaded!');
});
```

`decibelInsight('onCollectionChange', callback);`

This event will be triggered when the collection status of Decibel has changed.

Arguments

- `callback` (function): This callback will be passed the new and old status as parameters. It can contain any valid JavaScript code.

Example

```
// Log a message in the console whenever Decibel starts collecting data.
decibelInsight('onCollectionChange', function(newStatus, oldStatus) {
  if (newStatus === true) {
    console.log('Decibel is now collecting.');
  }
});
```

Collection Functions

These functions can be used to send additional information to Decibel.

`decibelInsight('formSubmitted', form);`

This function can be used to inform Decibel about a non-standard form submission, such as via AJAX.

Arguments

- `form` (Element|string): A HTML form element or CSS selector pointing to a form.

Examples

```
// Trigger submission of a form identified by DOM element.
decibelInsight('formSubmitted', form);

// Trigger submission of a form identified by a CSS selector.
decibelInsight('formSubmitted', '#contact-form');
```

`decibelInsight('sendApplicationError', message, [asSelector]);`

This function can be used to inform Decibel about application errors experienced by a visitor.

Arguments

- `message` (string): The error message shown to the user, or a CSS selector matching one or more elements containing error messages if the second parameter is used.
- `asSelector` (boolean): If set to `true`, the message parameter will be interpreted as a CSS selector and the text of one or more matching elements will be sent as error messages. Defaults to `false`.

Examples

```
// Pass a specific application error message to Decibel.
decibelInsight('sendApplicationError', 'Selected product is out of stock');

// Pass zero or more application error messages to Decibel
// by matching element in the HTML using a CSS selector.
decibelInsight('sendApplicationError', '.errorMessage', true);
```

decibelInsight('sendCustomDimension', dimension, value);

This function can be used to set custom information against a visitor, session or page view. The Custom Dimension must first be configured for the website through the Decibel App.

Arguments

- `dimension` (string): Name of the Custom Dimension.
- `value` (mixed): New value for the Custom Dimension. This value will override any value previously set for the specific Custom Dimension within its configured scope.

Examples

```
// Send Custom Dimension data with a fixed value.
decibelInsight('sendCustomDimension', 'loggedIn', true);

// Send Custom Dimension data retrieved from the JavaScript data layer.
decibelInsight('sendCustomDimension', 'customerType', dataLayer.customerType);
```

decibelInsight('sendGoal', name, [value], [currency]);

This function can be used to trigger goals. Goal value and currency can be sent also.

Arguments

- `name` (string): Name of the goal as configured in Decibel.
- `value` (float): Total monetary value for this goal (optional).
- `currency` (string): A three letter *ISO 4217* currency code (optional).

Examples

```
// Trigger a goal with no value attached.
decibelInsight('sendGoal', 'Add to Basket');

// Trigger a goal with a fixed value.
decibelInsight('sendGoal', 'Checkout', 250.56);

// Trigger a goal with a fixed value and currency.
decibelInsight('sendGoal', 'Checkout', 500, 'GBP');

// Trigger a goal with a value retrieved from the JavaScript data layer.
decibelInsight('sendGoal', 'Checkout', dataLayer.basketValue);
```

decibelInsight('sendHTTPError', code);

This function can be used to inform Decibel about HTTP server errors.

Arguments

- `code` (integer): Any valid HTTP status code in the 4xx or 5xx range. See *Status Code Definitions* for a complete list.

Example

```
// Inform Decibel that the current page
// is a 404 Page Not Found error page.
decibelInsight('sendHTTPError', 404);
```

decibelInsight('setPageRole', role);

This function can be used to specify the role of a page, allowing the creation of aggregated heatmaps and other reporting aggregations.

Arguments

- `role` (integer): ID of the role to be set for the page.

Example

```
// Inform Decibel that the current page is a PDP page.
decibelInsight('setPageRole', 4);
```

decibelInsight('trackCanvas', canvas);

Tracks the content of a canvas to ensure it is accurately reflected in the session replay. This function can be used when a large or heavily animated canvas is present on the page and automated tracking of the canvas is not desired.

Arguments

- `canvas` (Element|[Element]|string): A HTML canvas element, array of elements or a CSS selector pointing to one or more canvases.

Examples

```
// Track a canvas identified by DOM element.
decibelInsight('trackCanvas', canvas);

// Track a canvas identified by a CSS selector.
decibelInsight('trackCanvas', '#banner-animation');
```

`decibelInsight('trackPageView', [path], [params]);`

Sites that use JavaScript functionality to change page content without performing a page reload will need to use the Decibel JavaScript API to trigger page views.

Arguments

- `path` (string): Absolute path of the page (optional). If not provided, the path currently displayed in the browser will be used.
- `params` (object): An object containing one or more properties (optional). Parameters may include any of:
  - `title`: Title of the page. If not provided, the title currently displayed in the browser will be used.
  - `queryString`: The query string part of the URL for the page (for example "?page=2"). If not provided, the query string currently displayed in the browser address bar will be used.
  - `fragment`: The fragment part of the URL for the page (for example "#/Page/1"). If not provided, the fragment currently displayed in the browser address bar will be used.
  - `waitTime`: The time in milliseconds that it took the page or state to load.

Examples

```
decibelInsight('trackPageView', '/About-Us');
decibelInsight('trackPageView', '/Search', {
    queryString: '?page=1&q=Laptops',
    waitTime: 1000
});
```

`decibelInsight('updateUserId', userID);`

Provides a custom User ID for the current visitor. This is usually an ID generated from a website's login area and can be used to enable integration of Decibel data with existing internal systems such as CRM.

Arguments

- `userID` (string): This can be any text value up to 100 characters in length.

Example

```
// Assign a customer ID from the JavaScript data layer
// as the Decibel User ID.
decibelInsight('updateUserId', dataLayer.customerId);
```

Do not pass any personal details, such as the user's name or e-mail address, and avoid passing simple incrementing user IDs (for example, 12345). Best practice is to send a user ID hash using a *hash function* such as MD5 or SHA1.

Session Control Functions

The following functions provide granular control over the collection of sessions by Decibel.

`decibelInsight('endSession');`

Forces Decibel to stop collecting any further data for the current session.

Example

```
// Force Decibel to end the current session.
decibelInsight('endSession');
```

`decibelInsight('startSession', [newVisitor]);`

Forces Decibel to initiate a session and start collecting data. If a session is currently in progress, this will have no effect.

Arguments

- `newVisitor` (boolean): Whether to attribute the session to a new visitor profile (optional). If not provided, the new session will be attributed to the same visitor profile as the current session.

Example

```
// Force Decibel to start a session, for a new visitor profile.
decibelInsight('startSession', true);
```

`decibelInsight('restartSession', [newVisitor]);`

Forces Decibel to end the current session and start recording a new session. This is equivalent to calling the `decibelInsight('endSession')` and `decibelInsight('startSession')` functions consecutively.

Arguments

- `newVisitor` (boolean): Whether to attribute the session to a new visitor profile (optional). If not provided, the new session will be attributed to the same visitor profile as the current session.

Example

```
// Force Decibel to end the current session and start a new session.
decibelInsight('restartSession');
```

Integration Functions

The following functions support integration between Decibel and other analytics tools.

`decibelInsight('getLeadId');`

Returns Decibel Lead ID for the current session.

Returns

`(string)` Decibel Lead ID

Example

```
// Add the Decibel Lead ID to the JavaScript data layer.
dataLayer.leadId = decibelInsight('getLeadId');
```

`decibelInsight('getSessionId');`

Returns Decibel Session ID for the current session.

Returns

`(string)` Decibel Session ID

Example

```
// Add the Decibel Session ID to the JavaScript data layer.
dataLayer.sessionId = decibelInsight('getSessionId');
```

`decibelInsight('isCollecting');`

Determines if Decibel is successfully collecting data.

Returns

`(boolean)` Decibel collection status:

- `true` if Decibel is collecting data.
- `false` if no data is currently being collected.
- `null` if Decibel is not yet initialized.

Example

```
// Log a message in the console if Decibel is collecting data.
if (decibelInsight('isCollecting')) {
  console.log('Decibel is collecting');
});
```

Sampling Functions

The following functions provide control over the use of sampling within Decibel.

`decibelInsight('setCollection', status, [remember]);`

Enables or disables collection of data by Decibel. Calling this function will force the visitor into the collection sample if they are not already being tracked or remove them from the sample if they are currently being tracked.

Arguments

- `status` (boolean): New collection status for the current session.
- `remember` (boolean): Whether to remember the collection status for the next session (optional). If `false`, the status will apply for this session only. If `true` or omitted, the status will apply for this and all future sessions.

Examples

```
// Enable collection for the current session
// and future sessions for the current visitor.
decibelInsight('setCollection', true);

// Disable collection for the current session only.
decibelInsight('setCollection', false, false);
```

`decibelInsight('setRetention');`

Enables retention of data by Decibel if retention sampling is enabled. Calling this function will force the session into the retention sample if they are not already being retained. If retention sampling is not enabled, this function will have no effect.

Example

```
// Force the current session to be retained by Decibel.
decibelInsight('setRetention');
```

`decibelInsight('setFavorite', [retention]);`

Indicates that the session should be added as a favorite when it is processed.

Arguments

- `retention` (integer): Number of months for which the favorite will be retained. One of 1, 3, 6, 12 or 24. If omitted, the favorite will be retained indefinitely.

Example

```
// Mark the session to be added as a favorite.
decibelInsight('setFavorite');

// Mark the session to be added as a favorite and stored for 3 months.
decibelInsight('setFavorite', 3);
```

Session Replay Functions

These functions allow configuration of session replay recording throughout a visitor's journey.

`decibelInsight('pauseRecording', [delay]);`

Pauses the recording of visitor replay. This may be necessary before initiating a complex animation, to ensure that there is no reduction in framerate for a short period of time.

Arguments

- `delay` (integer): The number of milliseconds until recording will be resumed (optional). If not provided, recording will be paused until the `resumeRecording` end-point is called, or the visitor navigates to another page.

Examples

```
// Pause recording for the remainder of the current
// page view or until resumeRecording is called.
decibelInsight('pauseRecording');

// Pause recording for 500 millseconds.
decibelInsight('pauseRecording', 500);
```

`decibelInsight('resumeRecording');`

Resumes recording of visitor replay. This can be used if the `pauseRecording` end-point was called without a delay parameter, or to resume recording before the specific delay has passed. If called when recording is not paused, this end-point will have no effect.

Example

```
// Resume recording if it has previously been
// paused on the current page view.
decibelInsight('resumeRecording');
```

`decibelInsight('setFrameRate', rate);`

Sets the frame rate for visitor journey recordings.

Arguments

- `rate` (integer): Between 1 and 10 frames per second. The default frame rate is 5.

Example

```
// Reduce the frame rate to two frames per second.
decibelInsight('setFrameRate', 2);
```