Treasure Data JavaScript SDK API Version 3 Reference

The TD JS SDK is a set of tools, libraries, relevant documentation, and code samples to allow you to create Treasure Data functionality. This list is the set of building blocks that allow for the creation of that Treasure Data functionality.

Treasure(config)

Creates a new Treasure Data log instance. If the database does not exist and you have permissions, it will be created for you.

Parameters:

- **config** : Object (required) - instance configuration

Core parameters:

- **config.database** : String (required) - database name, must consist only of lower case letters, numbers, and _, must be longer than or equal to 3 chars, and the total length of database and table must be shorter than 129 chars.
- **config.writeKey** : String (required) - write-only key, get it from your user profile
- **config.pathname** : String (optional) - path to append after host. **Default: `/js/v3/events`**
- **config.host** : String (optional) - host to which events get sent. Default: `in.treasuredata.com`
- **config.development** : Boolean (optional) - triggers development mode which causes requests to be logged and not get sent. Default: `false`
- **config.logging** : Boolean (optional) - enable or disable logging. Default: `true`
- **config.globalIdCookie** : String (optional) - cookie td_globalid name. Default: `_td_global`
- **config.startInSignedMode** : Boolean (optional) - Tell the SDK to default to Signed Mode if no choice is already made. Default: `false`
- **config.jsonpTimeout** : Number (optional) - JSONP timeout (in milliseconds) Default: `10000`
- **config.storeConsentByLocalStorage** : Boolean (optional) - Tell the SDK to use localStorage to store user consent. Default: `false`

Track/Storage parameters:

- **config.clientId** : String (optional) - uuid for this client. When undefined it will attempt fetching the value from a cookie if storage is enabled, if none is found it will generate a v4 uuid
- **config.storage** : Object | String (optional) - storage configuration object. When `none` it will disable cookie storage
- **config.storage.name**: String (optional) - cookie name. Default: `_td`
- **config.storage.expires**: Number (optional) - cookie expiration in seconds. When 0 it will expire with the session. Default: `63072000` (2 years)
- **config.storage.domain**: String (optional) - cookie domain. Default: result of `document.location.hostname`

Server Side Cookie:

- **config.useServerSideCookie**: Boolean (optional) - enables/disable using ServerSide Cookie. Default: `false`
- **config.sscDomain**: String | () => String (optional) - Domain against which the Server Side Cookie is set. Default: `window.location.hostname`
- **config.sscServer**: String | (String) => String (optional) - hostname to request server side cookie from. Default: `ssc.${sscDomain}`

Personalization parameters

- **config.cdpHost**: String (optional) - The host to use for the Personalization API. Default: 'cdp.in.treasuredata.com'

Returns:

- Treasure logger instance object

Example:

```
var foo = new Treasure({
  database: 'foo',
  writeKey: 'your_write_only_key'
});
```

Treasure#addRecord(table, record, success, error)

Sends an event to Treasure Data. If the table does not exist it will be created for you.

Records will have additional properties applied to them if `$global` or table-specific attributes are configured using `Treasure#set`.

Parameters:

- **table**: String (required) - table name, must consist only of lower case letters, numbers, and _, must be longer than or equal to 3 chars, the total length of the database and table must be shorter than 129 chars.
- **record**: Object (required) - Object that will be serialized to JSON and sent to the server
- **success**: Function (optional) - Callback for when sending the event is successful
- **error**: Function (optional) - Callback for when sending the event is unsuccessful

Example:

```
var company = new Treasure({...});

var sale = {
  itemId: 100,
  saleId: 10,
  userId: 1
};

var successCallback = function () {
  // celebrate();
};

var errorCallback = function () {
  // cry();
}

company.addRecord('sales', sale, successCallback, errorCallback);
```

Treasure#fetchGlobalID(success, error, forceFetch, options)

If you set the `sameSite` value to `None`, the `Secure` property of the cookie will be set to true (it overwrites the `secure` option). More details at SameSite cookies.

Parameters:

- **success** : Function (optional) - Callback for when sending the event is successful
- **error** : Function (optional) - Callback for when sending the event is unsuccessful
- **forceFetch** : Boolean (optional) - Forces a refetch of global id and ignores cached version (default false)
- **options** : Object (optional) - Cookie options

Cookie options:

```
{
  path: '/',
  domain: 'abc.com',
  secure: true|false,
  maxAge: Number | String | Date,
  sameSite: 'None | Lax | Strict'
}
```

Example:

```
var td = new Treasure({...})

var successCallback = function (globalId) {
  // celebrate();
};

var errorCallback = function (error) {
  // cry();
}

td.fetchGlobalID(successCallback, errorCallback)

// with cookie options
td.fetchGlobalID(successCallback, errorCallback, false, {
  path: '/',
  secure: true,
  maxAge: 5 * 60 // 5 minutes,
  sameSite: 'None'
})
```

Treasure#fetchUserSegments(options, success, error)

This funcrtion is not enabled by default.

Parameters:

- **options**: Object (required) - User Segment object
    - **options.audienceToken**: String or Array (required) - Audience Token(s) for the userId
    - **options.keys**: Object (required) - Key Value to be sent for this segment
- **success**: Function (optional) - Callback for receiving the user key and segments
- **error**: Function (optional) - Callback for when sending the event is unsuccessful

Example:

```
var td = new Treasure({...})

var successCallback = function (values) {
  /* values format => [... {
    key: {
      [key]:value
    },
    values: ["1234"],
    attributes: {
      age: 30
    },

  } ... ]*/
  // celebrate();
};

var errorCallback = function (error) {
  // cry();
};

td.fetchUserSegments({
  audienceToken: ['token1', 'token2'],
  keys: {
    someKey: 'someValue',
    someOtherKey: 'someOtherValue',
  }
}, successCallback, errorCallback)
```

Treasure#blockEvents

Block all events from being sent to Treasure Data.

Example:

```
var td = new Treasure({...})
td.trackEvent('customevent')
td.blockEvents()
td.trackEvent('willnotbetracked')
```

Treasure#unblockEvents

Unblock all events; events will be sent to Treasure Data.

Example:

```
var td = new Treasure({...})
td.blockEvents()
td.trackEvent('willnotbetracked')
td.unblockEvents()
td.trackEvent('willbetracked')
```

Treasure#areEventsBlocked

Informational method, expressing whether events are blocked or not.

Example:

```
var td = new Treasure({...})
td.areEventsBlocked() // false, default
td.blockEvents()
td.areEventsBlocked() // true
```

Treasure#setSignedMode

Permit sending of Personally Identifying Information (PII) over the wire: td_ip, td_client_id, and td_global_id

Example:

```
var td = new Treasure({...})
td.setSignedMode()
td.trackEvent('willbetracked') // will send td_ip and td_client_id; td_global_id will also be sent if set.
```

Treasure#setAnonymousMode

Prohibit sending of Personally Identifying Information (PII) over the wire: td_ip, td_client_id, and td_global_id

Parameters:

- **keepIdentifier**: Boolean (optional) - Keep the cookies

By default `setAnonymousMode` will remove all cookies that are set by Treasure Data JavaScript SDK, you can set `keepIdentifier` parameter to `true` t o not remove the cookies.

Example:

```
var td = new Treasure({...})
td.setAnonymousMode()
td.trackEvent('willbetracked') // will NOT send td_ip and td_client_id; td_global_id will also NOT be sent if
set.
```

Treasure#inSignedMode

Informational method, indicating whether `trackEvents` method will automatically collect td_ip, td_client_id, and td_global_id if set.

Example:

```
var td = new Treasure({...})
td.inSignedMode() // false, default
td.trackEvent('willbetracked') // will NOT send td_ip and td_client_id; td_global_id will also NOT be sent if
set.
td.setSignedMode()
td.inSignedMode() // true
td.trackEvent('willbetracked') // will send td_ip and td_client_id; td_global_id will also be sent if set.
```

Treasure#fetchServerCookie(success, error, forceFetch)

This functionality complies with ITP 1.2 tracking. Contact customer support to enable this feature.

Parameters:

- **success**: Function (optional) - Callback for when sending the event is successful
- **error**: Function (optional) - Callback for when sending the event is unsuccessful
- **forceFetch**: Boolean (optional) - Forces a refetch of server-side id and ignores cached version (default false)

Example:

```
var td = new Treasure({...})

var successCallback = function (serverSideId) {
  // celebrate();
};

var errorCallback = function (error) {
  // cry();
}

td.fetchServerCookie(successCallback, errorCallback)
```

Treasure#resetUUID

Parameters:

- **suggestedStorage**: Object (optional) - Custom storage configuration
- **suggestedClientId**: String (optional) - Custom client id

Reset the client's UUID, by setting Treasure Data for the `td_client_id`. You can specify custom storage and custom client id.

See the **Track/Storage parameters** section for more information on storage's configuration.

Example:

```
var td = new Treasure({...})
td.resetUUID() // set td_client_id as random uuid
var td = new Treasure({...})
td.resetUUID(
  {
    name: '_td', // cookie name
    expires: 63072000,
    domain: 'domain',
    customDomain: true/false
    path: '/'
  },
  'xxx-xxx-xxxx' // client id
)
```

Treasure#trackClicks

Setup an event listener to automatically log clicks. The event is hooked similar to the following:

- `role=button` or `role=link`
- `<a>`
- `<button>`
- `<input>`). exclude for `<input type='password'>`

Example:

```
var td = new Treasure({...})
td.trackClicks({
    element         : '...'
    extendClickData : '...'
    ignoreAttribute : '...'
    tableName       : '...'
    })
```

- element: HTMLElement -> Default is `window.document`. The default setting will observe all elements above. You can set an element if you want to focus on a particular element.
- extendClickData: Function -> Default is function to set element attributes. You can set function adding special tracking data by extending `function(e: event, elementData: ElementObject)`.
- ignoreAttribute: string -> Default is `"td-ignore"` You can set the attribute name to ignore element. (e.g. `<span role='button' class='button-design' id='button-id' td-ignore />`)
- tableName: string -> Default tableName is `"clicks"`. Click tracking event will be stored into `tableName` in TreasureData.

Treasure#trackPageview(table, success, error)

Helper function that calls trackEvent with an empty record.

Parameters:

- **table**: String (required) - table name, must be between 3 and 255 characters and must consist only of lower case letters, numbers, and _
- **success**: Function (optional) - Callback for when sending the event is successful
- **error**: Function (optional) - Callback for when sending the event is unsuccessful

Example:

```
var td = new Treasure({...});
td.trackPageview('pageviews');
```

Treasure#trackEvent(table, record, success, error)

Creates an empty object, applies all tracked information values and applies record values. Then it calls `addRecord` with the newly created object.

Parameters:

- **table**: String (required) - table name, must be between 3 and 255 characters and must consist only of lower case letters, numbers, and _
- **record**: Object (optional) - Additional key-value pairs that get sent with the tracked values. These values overwrite default tracking values
- **success**: Function (optional) - Callback for when sending the event is successful
- **error**: Function (optional) - Callback for when sending the event is unsuccessful

Example:

```
var td = new Treasure({...});

td.trackEvent('events');
/* Sends:
{
  "td_ip": "192.168.0.1",
  ...
}
*/

td.trackEvent('events', {td_ip: '0.0.0.0'});
/* Sends:
{
  "td_ip": "0.0.0.0",
  ...
}
*/
```

Treasure#set()

Default value setter for tables. Set default values for all tables by using `$global` as the setter's table name.

Treasure#set(table, key, value)

Useful when you want to set a single value.

Parameters:

- **table** : String (required) - table name
- **key**: String (required) - property name
- **value** : String | Number | Object (required) - property value

Example:

```
var td = new Treasure({...})
td.set('table', 'foo', 'bar');
td.addRecord('table', {baz: 'qux'});
/* Sends:
{
  "foo": "bar",
  "baz": "qux"
}
*/
```

Treasure#set(table, properties)

Useful when you want to set multiple values.

Parameters:

- **table** : String (required) - table name

- **properties**: Object (required) - Object with keys and values that you wish applies on the table each time a record is sent

Example:

```
var td = new Treasure({...})
td.set('table', {foo: 'foo', bar: 'bar'});
td.addRecord('table', {baz: 'baz'});
/* Sends:
{
  "foo": "foo",
  "bar": "bar",
  "baz": "baz"
}
*/
```

Treasure#get(table)

Takes a table name and returns an object with its default values.

Only available once the library has loaded. Wrap any getter with a `Treasure#ready` callback to ensure the library is loaded.

Parameters:

- **table** : String (required) - table name

Example:

```
var td = new Treasure({..});
td.set('table', 'foo', 'bar');
td.get('table');
// {foo: 'bar'}
```

Treasure#ready(fn)

Takes a callback which gets called once when the library and DOM have both finished loading.

Parameters:

- **fn**: Function (required) - callback function

```
/* javascript snippet here */
var td = new Treasure({...})
td.set('table', 'foo', 'bar');

td.ready(function(){
  td.get('table');
  // {foo: 'bar'}
});
```