



BehavioSec Documentation

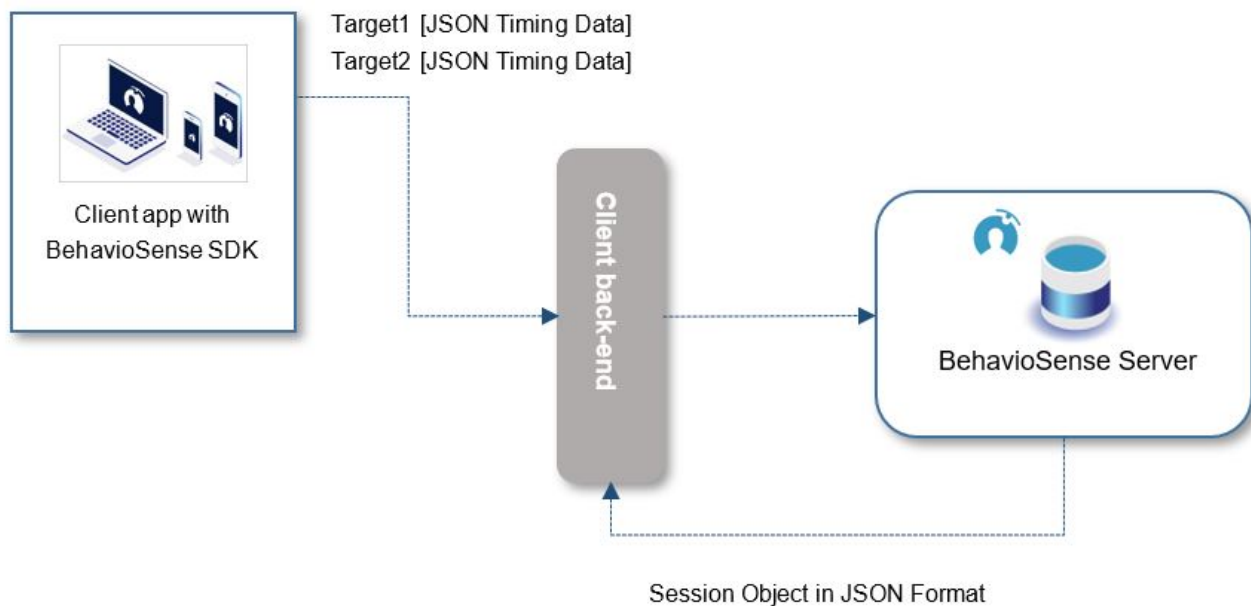
[Contact Support for questions.](#)

Table of Contents

BehavioSec Documentation	1
Understanding User Session Data	5
<i>Getting Familiar with Session Data</i>	5
<i>Session Object</i>	6
Transaction Object	7
Target Object	8
<i>Understanding the Timing Data</i>	8
Timing Data Format Identifiers	10
Normal Collection Mode (Timing Data Format f)	10
Anonymous Collection (Timing Data Format fa)	11
Data Integrity Mode (Timing Data Format w)	11
Mouse and Touch for JavaScript (Timing Data Format c)	12
Browser Metadata (Timing Data Format m)	13

Understanding User Session Data

Session data is comprised of all the collected and analyzed data from a user session. It includes data from a session as a whole, data from the transaction and target, the behavioral timing data, and other related data. The BehaviorSec SDK collects data from a customer application and sends it through a client's backend to the BehaviorSense Server. BehaviorSense performs its analysis and returns the data in a JSON-formatted Session object. The diagram below depicts this process.



A user session is usually, but not necessarily, associated with a client application's defined session ID. BehaviorSense uses its own internal sessionId.

BehaviorSense generates new session data for each new transaction, and updates it with consecutive transactions with the same sessionId. A session with several transactions is re-evaluated upon each new transaction.

BehaviorSense uses the session data to validate a user's entire session by comparing how similar it is to the [User Profile](#) over the course of several transactions. If the session is not similar enough, it is considered to be a rejected session, and the behavioral timing data is not added to the User Profile.

Getting Familiar with Session Data

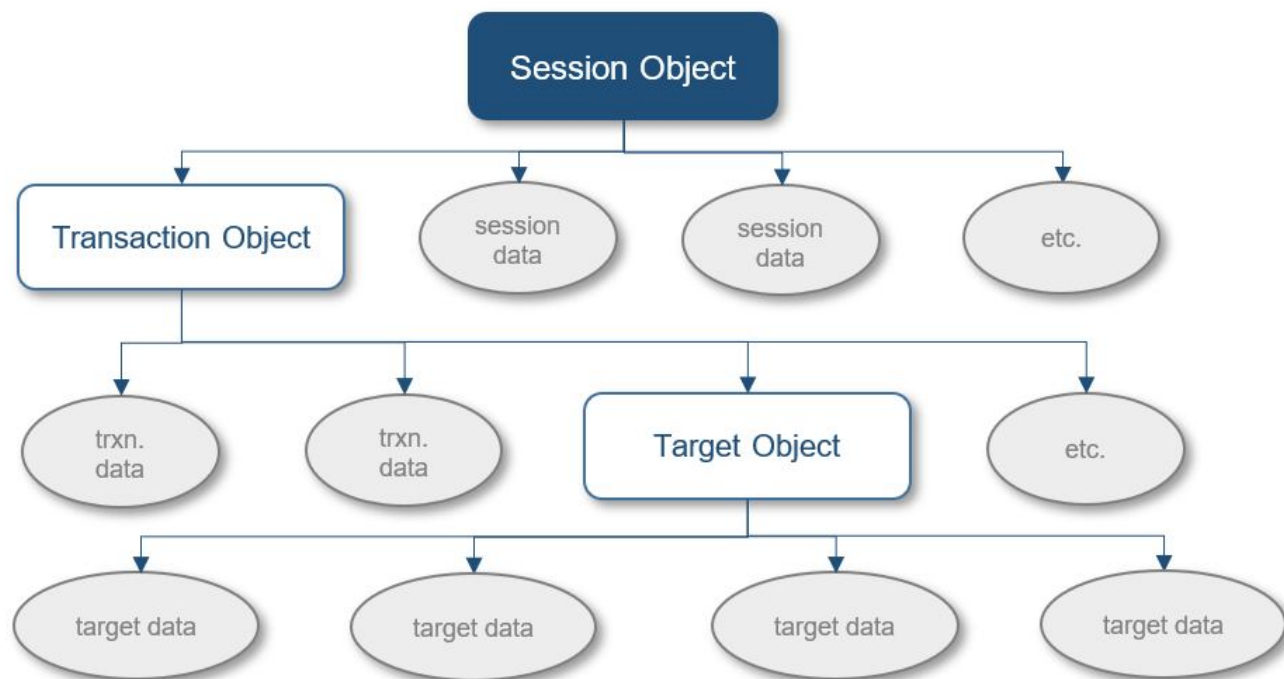
The raw, or initial, session data is stored in the BehaviorSense database. After BehaviorSense performs analysis on the raw data, it returns modified data in the API calls that return the results of sessions ([BindJourney](#), [GetInvestigationReport](#), [GetReport](#), [GetReportAndInvestigate](#), and [GetSession](#)).

The BehaviorSense API returns data for three major objects:

- [Session](#)
- [Transaction](#)
- [Target](#)

Each object contains other objects however this page focuses on the session, transaction, and target objects. There are many types of data reported for a session and they are listed at [Return Data Format](#), along with which objects can contain them.

The diagram below demonstrates how these objects relate to one another, with the session object at the top. The circles labelled *data* represent the various data types that are reported for each object.



Session Object

The Session object contains all the data that is collected and reported during a session. One Session object can contain one or more Transaction objects. There are many types of data reported for a session and they are listed at [Return Data Format](#), along with which objects can contain them.

The Dashboard displays the data within the returned Session object using graphs, tables, and lists.

Exporting the Raw Session Data for Troubleshooting

For Support purposes, you can see and export the raw data in the Dashboard [Data > Session Log > Session Details > Session Data](#) panel. Click View Session Data to see the raw session data.

1
Go
Import...
Export...

	User ID	Session ID	Risk	Score	Start Date	Options
✓	desktop-62d0etl.user	0a92db62-d3e5-43b2-bf93-57d694a8c33d.1553243697,25796.behaviocredential	6.94	99.95	2019-03-22 09:34:57	^ ~ Q

Audit for this User
Link to this User
Link to this Session
All Sessions for this User

Add to Favorites
View Session Data
Export Session Data
Force Train Session
Remove Session

Session Data

Add to Favorites

View Session Data

Export Session Data

Force Train Session

Remove Session

Click to view session data

Session Details

User ID: desktop-62d0etl.user

Session ID: 0a92db62-d3e5-43b2-bf93-57d694a8c33d.1553243697,25796.behaviocredential

Notes: Timing data does not contain collector header.

Green

- Score 99.95
- Confidence 99.50
- Recognition Ratio 100.00
- Risk 6.94

Date	Device	IP Address
Start: 2019-03-22 09:34:57	DESKTOP	fe80::fc07:624:1e70:53f8
End: 2019-03-22 09:34:57		
Duration: (hh:mm:ss) 00:00:00		
Finalized: 2019-03-22 09:34:54		

Export

Flags

📄
Data Integrity

📄
Page Definitions

🛡️
Bot Detection

👤
User Integrity

⚠️
Usage Anomaly

🔄
IP Changed

🖨️
Device Changed

🗑️
Data Corrupted

🔥
Session Corrupted

🔄
Replay Attack

🖨️
Remote Access

📄
New Sub Profile

🖨️
Device Reputation

🔒
Origin Hidden

✅
White Listed

?

Transactions During this Session

	Transaction ID	Risk	Score	Conf	Recog	Date
✓	0a92db62-d3e5-43b2-bf93-57d694a8c33d.1553243697,25796.behaviocredential:0	6.94	99.95	99.50	100.00	2019-03-22 09:34:57

Click Export Session Data to export a .txt file.

See [Quick Actions](#).

Transaction Object

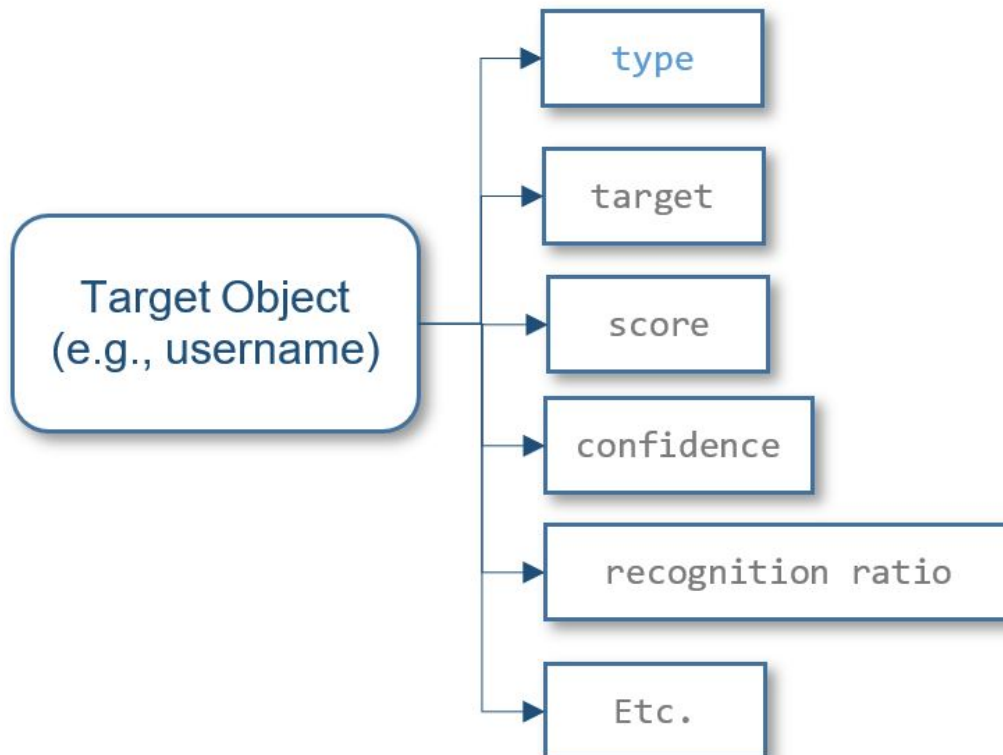
One or more Transaction objects are within a Session. Transactions appear within the data key as an array. A Transaction has its own Behavioral Score, Confidence, and Recognition Ratio, plus other data. See [Transaction Object](#).

Target Object

The Target object is found within the Transaction as the targets key. It contains its own Behavioral Score, Confidence, and Recognition Ratio. See [Target Object](#).

A target in BehaviorSense is a specific keyboard input to a field, mouse interaction, or other input in a client application where timing data is collected. In the JSON results, a Target object contains many variables such as:

- type: the meta data type (e.g., f, fa, fm, w, c, or m)
- target: the string that is used to identify it (e.g., text#username)
- score
- confidence
- etc.



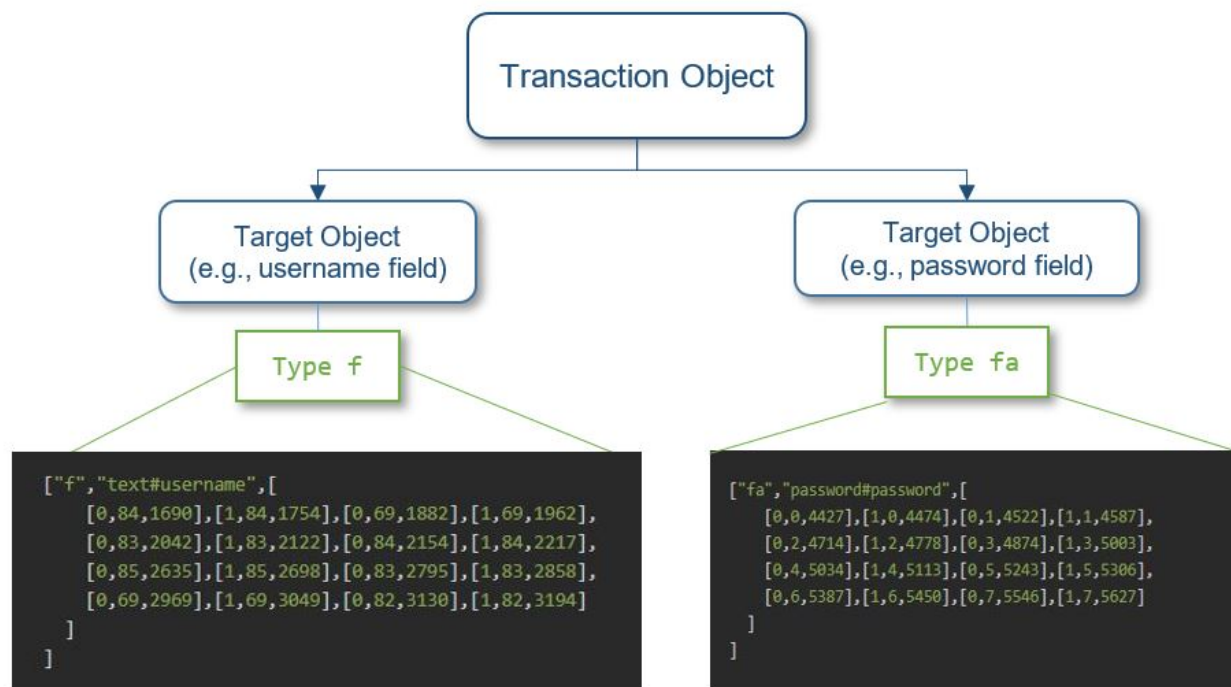
Understanding the Timing Data

Timing data is the information collected by a BehaviorSec SDK for every registered target. The timing data is important because it is the data used by BehaviorSense to analyze behavior. An SDK collects timing data from a particular web page or mobile application. As the user interacts with the targets, the SDK collects the timing data.



Activities such as the timing between key presses and how the user interacts with the mouse or touch screen are recorded as timing data. The timing data is formatted in JSON as an array and it lies within the target object.

An SDK collects timing data for each target object. The data is transmitted, usually by a backend web server or proxy, to the BehaviorSense Server in JSON format using an array. Timing data is stored at the target object level. In the diagram below, the transaction object has two targets of type f and fa, but other types of targets can be related.



Timing Data Format Identifiers

BehavioSense classifies the targets using the following data collection mode timing data format identifiers:

- **f**: Normal, keypress and keystroke timing data
- **fa**: Anonymous, keystroke timing data for anonymous fields (recommend using Anonymous Masked instead, see [Normal vs. Anonymous Modes](#))
- **fg**: Free-form Gesture, swipe or press data for mobile touch, iOS and Android SDKs
- **fm**: Anonymous Masked, keystroke timing data for anonymous masked fields
- **w**: Data Integrity Metadata, number of keys in the target and the path to the target
- **c**: Browser Mouse/Touch, mouse movement and touch timing data for browsers, JavaScript SDK
- **m**: Metadata, navigator, screen objects, and versioning data
- **mi**: Metadata about the mobile device
- **b**: Mobile metadata related to versioning (*New in 5.2*)
- **bi**: Mobile metadata related to versioning (*New in 5.2*)
- **th**: Mobile metadata related to future features (*New in 5.2*)

When you see target objects in the session data, they are associated with these type identifiers. See [Timing Data Format](#).

Normal Collection Mode (Timing Data Format f)

Keystroke data, including keystrokes, is identified as f. An example of type f is given below.

```
[ "f", "text#username", [  
  [0,84,1690], [1,84,1754], [0,69,1882], [1,69,1962],  
  [0,83,2042], [1,83,2122], [0,84,2154], [1,84,2217],  
  [0,85,2635], [1,85,2698], [0,83,2795], [1,83,2858],  
  [0,69,2969], [1,69,3049], [0,82,3130], [1,82,3194]  
]  
]
```

This type of data uses the following format:

```
[ "f", "targetIdentifier", [ [event 1], [event 2], [event n] ] ]
```

The first value is the type. In our example it is f which means it is a basic keystroke field. Next is the target identifier which is completely unique to your system. This represents the name given to the target field. The example we give has text#username as the identifier. The last part [event 1] is the timing data or array.

The timing array uses the following format:

```
[down/up, key, timestamp]
```

- 0 represents a downward keystroke and 1 represents an upward keystroke.
- Key is the keycode of the button that triggered the event.
- Timestamp is the time in milliseconds which is used to get the delta between events.

Anonymous Collection (Timing Data Format fa)

Important: BehaviorSec recommends that new implementations use Anonymous Masked collection mode instead of fa. Anonymous Masked Keystroke mode (fm) is available with JavaScript SDK Version 261, iOS SDK 2.0.2, Android SDK 1.9.5, and BehaviorSense Version 5.1 or newer.

See [Normal vs. Anonymous Collection Modes of Keystroke Data](#).

Anonymous keystroke data is classified as fa. An example of type fa is given below.

```
[ "fa", "password#password", [  
  [0,0,4427],[1,0,4474],[0,1,4522],[1,1,4587],  
  [0,2,4714],[1,2,4778],[0,3,4874],[1,3,5003],  
  [0,4,5034],[1,4,5113],[0,5,5243],[1,5,5306],  
  [0,6,5387],[1,6,5450],[0,7,5546],[1,7,5627]  
]  
]
```

This type of data uses the following format:

```
[ "fa", "targetIdentifier", [ [event 1], [event 2], [event n] ] ]
```

The first item is the type. In our example, it is fa which means it is a basic keystroke field that is anonymous. A form field using the anonymous data format will no longer track which keys that are typed. Next is the target identifier which is completely unique to your system. This represents the name given to the target field. The example we give has password#password as the identifier. The last part [event 1] is the timing data or array. The timing array uses the following format:

```
[down/up, index, timestamp]  
* 0 represents a downward keystroke and 1 represents an upward keystroke.  
* Index is the index of the button that triggered the event.  
* Timestamp is the time in milliseconds which is used to get the delta between events.
```

Data Integrity Mode (Timing Data Format w)

Data integrity is classified as type w. An example of type w is given below.

```
[
  "w",
  [
    {
      "text#username": 8
    },
    {
      "password#password": 8
    }
  ],
  "/BehavioSenseDemo/Login/"
]
```

This type of data uses the following format:

```
["w", [ {target 1}, {target 2}, {target n} ], view identifier]
```

The first item is the type. In our example, it is w which means it is a data integrity type. Next is the target which is a combination of the name and value. The last item is called the view identifier which is a unique key that provides the path.

The target format is:

```
{"targetId": char_count}
* targetId is the target identifier or name.
* The char_count is the number of inputted characters at the time of submit.
```

Mouse and Touch for JavaScript (Timing Data Format c)

Mouse and touch data are classified as type c. An example of type c is given below.

```
[
  "c",
  [
    ["v", 1427, 837, 585],
    ["mm", 191, 0, 583],
    ["mm", 191, 1, 586],
    ["mm", 190, 4, 586],
    ["t", "FIELDSET##FORM#demoform", 589],
    ["mm", 188, 8, 588],
    ["mm", 187, 12, 591],
    ["mm", 187, 16, 592],
    ["mm", 185, 20, 594],
    ["mm", 185, 24, 597],
    ["mm", 184, 28, 599],
  ]
]
```

```
[ "mm", 184, 32, 601 ],  
[ "mm", 184, 35, 603 ],  
[ "mm", 184, 38, 604 ],  
[ "md", 538, 39, 610, 0 ],  
[ "mu", 538, 39, 959, 0 ]  
],  
"/BehavioSenseDemo/Login/"  
]
```

This type of data is given the following format:

```
["c", [ [event 1], [event 2], [event 3] ], view identifier]
```

The first value is the code for type c or mouse/touch. The next item is the [event], which is an array that contains:

```
[type, .. data ...]
```

The type can be:

- mm for movement
["mm", x_pos, y_pos, timestamp]
- v for view port
["v", width, height, timestamp]
- md for mouse down
["md", x_pos, y_pos, timestamp, button_id]
- mu for mouse up
["mu", x_pos, y_pos, timestamp, button_id]
- t for move over component
["t", component_identifier, timestamp]

The last item is the unique view identifier or path. For more details see [Type c](#).

Browser Metadata (Timing Data Format m)

This type of data contains information extracted from the browser's JavaScript navigator and screen objects (if applicable). An example of type m for a navigator object is shown below.

```
[  
  "m",  
  "n",  
  {  
    "vendorSub": "",  
    "productSub": "20030107",  
    "vendor": "Google Inc.",  
    "maxTouchPoints": 0,  
  }  
]
```

```
"hardwareConcurrency": 8,
"appCodeName": "Mozilla",
"appName": "Netscape",
"appVersion": "5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/50.0.2661.94 Safari/537.36",
"platform": "Win32",
"product": "Gecko",
"userAgent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36",
"language": "sv",
"languages": [
  "sv-SE",
  "sv",
  "en-US",
  "en"
],
"onLine": true,
"cookieEnabled": true,
"doNotTrack": null,
"geolocation": {},
"webkitTemporaryStorage": {},
"webkitPersistentStorage": {},
"serviceWorker": {
  "controller": null
},
"mediaDevices": {},
"permissions": {},
"presentation": {}
}
]
```

The m type data is formatted as:

```
["m", subtype, data]
```

The first item is type m. The subtype is type of meta data such as n (navigator), s (screen), and v (version).

- navigator_object contains information about the browser. (The information provided varies between browsers.)
- screen_object contains information about the screen. (The information provided varies between browsers.)
- version is an integer describing the version of the data collector.

The last item is data that contains all the data associated with the type and subtype. For more detailed information, see [Type m](#).

