Client Framework Data Integration Guide

Contents

IBM Tealeaf CX Client Framework Data Integration Guide

This guide provides information on the Tealeaf CX features that you use when you integrate your android, iOS, and webview applications in Tealeaf. The guide provides tasks and information on the events and hit attributes that you configure to log mobile session data.

Work with client framework data in Tealeaf

After you integrate one of the Tealeaf® client frameworks with your web application or mobile native application, you configure Tealeaf to properly capture and process the data. You configure data privacy, capture, events. The Canister, and TLI server.

The configuration process to work with client framework data in Tealeaf includes:

1. Configure data privacy for client frameworks
2. Install and implement the Tealeaf framework for capture
3. Configure Canister and TLI Server, if necessary
4. Configure Tealeaf Events

Configure data privacy for client frameworks

Configure data privacy rules to mask any data captured from the client frameworks, such as passwords, that must not be submitted to Tealeaf for capture.

As part of any implementation through Tealeaf Professional Services, data privacy is configured and applied for any data that is submitted from a monitored client application. Before you enable capture of client framework data, verify that privacy configuration was properly enabled and configured.

Install and implement the Tealeaf framework for capture

After you integrated the framework with your application, you must configure the framework to capture the data and configure Tealeaf and process the data.

Use the information in this guide...	To install and implement data capture for...
IBM® Tealeaf CX Mobile Android Logging Framework Guide	native Android applications and for hybrid native Android and WebView applications.
IBM Tealeaf CX Mobile iOS Logging Framework Guide	native iOS application and for hybrid native iOS and WebView applications.
IBM Tealeaf CX UI Capture j2 Guide	JavaScript WebView applications.

Other Tealeaf configuration

You might need to configure the Canister and TLI Server for your deployment:

Tealeaf area	Description	Applicability
Canister Safety Limits	To prevent runaway growth of active sessions in the Canister, Tealeaf imposes maximum limits on duration, data size, and number of hits for a session. If any of these limits is exceeded, the session is automatically	The Canister Safety limits apply to client framework sessions, including sessions from mobile native applications. • For mobile native application sessions, data is submitted in

Tealeaf area	Description	Applicability
	closed in the Canister, even if the visitor is still using the application. Subsequent hits are recorded to a new session.	compressed format to the PCA, which writes values for the request and response data size before it was uncompressed. In some cases, the uncompressed text data can be 8x larger. – If you enabled screen capture through your client framework, this data is submitted in a compressed file format (png) and does not expand inside Tealeaf. • Depending on the volume of traffic, you might want to revisit the Canister Safety Limits.
TLI Server	A TLI server can be optionally deployed to store static content, such as images, style sheets, and JavaScript, onto a separate server. The TLI server can also manage versioning of this static content so that the original version of static content is available for replay and auditing purposes indefinitely. • When a TLI server is deployed, the TLI session agent scans hits based on MIME type. The agent stores content onto the server that is based on a configured set of types.	For mobile native application sessions captured through a Logging Framework, TLI does not apply, since any captured screen captures are embedded in the session data. The captures are not referenced by URL to an origin server. Therefore, TLI cannot be used to store versions of this content. TLI servers might be used with mobile web deployments.

Configure Tealeaf Events

There are three things to keep in mind about events:

- Tealeaf provides a set of events and event-related objects for use in capturing and processing data from client frameworks.
- Data that is submitted from a Tealeaf client framework is in the form of JSON-based messages. These messages are processed and stored as individual steps in the session data. The step-based events and hit attributes that you create from these messages perform like other Tealeaf objects, but they are configured in a different manner.
- You might find it useful to work through a complete scenario in which you create step-based events and hit attributes from client framework data and then integrate those event objects into your data from your browser-based web application.

Client framework versions and licenses

Minimum required versions

This documentation applies to the these versions of Tealeaf software licenses:

License	Minimum version
IBM Tealeaf cxImpact	Release 8.5 or later
PCA	PCA Build 3330

Earlier versions

If your system does not meet the minimum requirements, JSON-based processing within Tealeaf and step-based eventing are not supported.

Refer to this documentation for the earlier versions of the Tealeaf client frameworks.

- IBM Tealeaf CX UI Capture for AJAX Guide
- Android Logging Framework Reference Guide
- iOS Logging Framework Reference Guide

Required licenses

To use any Tealeaf client framework, you must have the IBM Tealeaf cxImpact license.

IBM Tealeaf CX Mobile license is required for Android Logging Framework and iOS Logging Framework.

If you do not have CX Mobile

If you did not license IBM Tealeaf CX Mobile and you have the UI Capture j2 Framework, you can not display mobile-specific features and events during replay.

Versions of UI Capture for Ajax

Depending on the licenses that you acquired, the following versions of UI Capture are available and determine the method by which data is captured and submitted to Tealeaf for processing. These licenses and their supported methods also determine the type of applications that IBM Tealeaf CX UI Capture for AJAX is able to monitor.

IBM Tealeaf CX UI Capture for AJAX is only available to legacy users. New users must use IBM Tealeaf UI Capture

For Release 8.5 and later, the JSON version of IBM Tealeaf CX UI Capture for AJAX is the only one that is supported. For more information about support for the legacy XML version, contact Tealeaf http://support.tealeaf.com.

License	Method of Capturing and Submitting	Monitored applications
IBM Tealeaf CX only	JSON	fixed desktop web applications
IBM Tealeaf CX and IBM Tealeaf CX Mobile	JSON	fixed desktop web applications mobile web browser applications

Versions of Logging Frameworks

For the Tealeaf Logging Frameworks, the following versions are available for this release. Logging Frameworks require IBM Tealeaf CX Mobile license:

Depending on the licenses that you acquired, the following versions of the Tealeaf Logging Frameworks are available and determine the method by which data is processed within Tealeaf.

License	Method of Processing Hits	Description
IBM Tealeaf CX Mobile only	Hit-splitting	When data captured from the Logging Frameworks is processed in the Windows pipeline, messages are split into separate, individual hits and added to the session data. Hit-splitting is considered the legacy method of processing client framework data. It is likely to be deprecated in a future release. This method requires the deployment of a specific session agent in the Windows pipeline to split hits
IBM Tealeaf CX and IBM Tealeaf CX Mobile	JSON	Data is captured from the client frameworks and submitted in a compact JSON format. This data is made available in a more readable format through Browser-Based Replay, from which you can create step-based events and attributes for tracking purposes. This method is the current version of processing client framework data and is available in Release 8.5 or later. No additional configuration is required.

Data privacy in Tealeaf client frameworks

Tealeaf client frameworks provide multiple security features to ensure that sensitive application and user data is safeguarded for transport or retained only in the client application. Using controls that you can configure in each client framework, you can define the specific set of data that is blocked or masked from transport to Tealeaf.

Sensitive data that was cleansed through a client framework never reaches Tealeaf, which ensures that your customer's interactions are secure.

- Masks can be expressed as explicit strings, replacements for character types, or custom functions.
- Extra security features.

As part of any implementation through Tealeaf Professional Services, data privacy is configured and applied for any data that is submitted from a monitored client application. Before you enable capture of client framework data, verify that privacy configuration was properly enabled and configured.

If you have questions about how to implement data privacy in UI Capture, contact Tealeaf Professional Services.

Data Masking and Blocking for Client Frameworks

Do not copy or transmit sensitive data when not necessary. To protect your users' privacy, do not log sensitive data like passwords and credit card numbers. You can see the format this sensitive data without seeing the data when troubleshooting. Removing the content of this information is called **masking**.

In Release 8.5, Tealeaf introduces step-based eventing, which simplifies and unifies event capture from all client frameworks and enhances performance. Because of changes in how the data is bundled, Tealeaf recommends applying data privacy through the individual client frameworks, instead of using the Tealeaf server methods for data privacy.

Limitations

Any data that is masked or blocked by a Tealeaf Client Framework is never available within Tealeaf for processing, search, and reporting.

For data submitted from a client framework in JSON format, privacy must be applied by using the client framework's controls. Application of privacy controls to JSON-based data after it is submitted to Tealeaf requires advanced abilities in Tealeaf privacy controls and regular expression development.

General security features

Security features apply to communications with the target page, local storage cache, logging levels, data masking, and data blocking.

HTTPS communications with target page

Client frameworks provide a configurable means of submitting captured events to the Tealeaf target page by HTTPS.

There is a performance overhead in sending data over HTTPS. The performance impacts depend on the type of application, the network bandwidth, and the network load.

To deploy HTTPS communications, you might deploy any of the provided Tealeaf target pages and configure on the server the directory permissions for HTTPS access.

This table lists what you do to deploy HTTPS communications:

Client Framework	What you do
UI Capture	To configure submitting by HTTPS for AJAX, you must configure the `tlsecureurl` property for the Tealeaf target page. No configuration is needed for UI Capture j2. The server-side endpoint can manage both secure and non-secure POSTs. The protocol of the POST is determined by the protocol of the parent page.
Android Logging	To configure HTTPS submissions, configure the `PostMessageUrl` property in the `TLFConfigurableItems.properties` file to use the `https://` protocol.
iOS Logging Framework	To configure HTTPS submissions, configure the `PostMessageUrl` property in the `TealeafBasicConfig.plist` file to use the `https://` protocol.

Disabling local storage cache

If needed, some client frameworks can be configured to disable the local storage cache, which is used to gather client events for submission to Tealeaf.

During cached operations, if the client framework is unable to connect to the Tealeaf target, events are queued in a local memory buffer. If the buffer fills before the connection is restored, the last event that is stored in the buffer is discarded.

- This buffer is cleared on restart or power down.

When the local storage cache is disabled, client events are sent as soon as they are detected by the client framework. Since there is a data overhead for submitting a package of events, the data overhead increases when sending a single event per package, which might affect client and network performance.

This table lists what you do to disable the local storage cache:

Client Framework	What you do
UI Capture	For AJAX, the local storage cache size and availability can be configured through the user's browser. UI Capture j2 does not support cached operations and requires a live connection to the web server to submit POSTs. If a live connection to the web server that hosts the Tealeaf target page is not available, captured UI data is discarded.
Android Logging	For the Android framework, you can configure local cache settings, including disabling it altogether using the `HasToPersistLocalCache` setting.
iOS Logging Framework	For the iOS framework, you can configure local cache settings, including disabling it altogether using the `HasToPersistLocalCache` setting.

Data masking

Each client framework enables the masking of sensitive data on the client before it is submitted to Tealeaf. Through configuration, you can identify for the client framework the objects in the monitored application that must be masked. For example, you can indicate that each numeral in a credit card field must be replaced by an X:

```
credit_card_num=XXXX-XXXX-XXXX-XXXX
```

This table lists what you do to configure data masking:

Client Framework	What you do
UI Capture	For AJAX, in the `TealeafClientCfg.js` file, you can specify the fields to mask with the `tlFieldBlock` array. JSON message fields can be specified by name, ID, or class name. For UI Capture j2, use the UIC Configuration Wizard to specify the user input fields to mask. Input fields can be specified by HTML ID, name, CSS class, or any custom attribute. If a live connection to the web server that hosts the Tealeaf target page is not available, captured UI data is discarded.
Android Logging	Through the `TLFConfigurableItems.properties` file, you can configure the fields to mask, the masking characters, and other features.
iOS Logging Framework	Through the `TealeafBasicConfig.plist` file, you can configure the fields to mask, the masking characters, and other features.

Data blocking

Data blocking removes all values that are related to the blocked field.

As needed, you can configure the data for individual items to be blocked altogether. If the data in the data masking example was blocked, the submitted item would be:

```
credit_card_num=
```

JSON-based values are not stored in name/value format. As a result, the configuration that is required to block JSON-based data is different. In the example below, the value of the textbox (`myLoginID`) must be masked:

```
{
                    "type": 4,
                    "offset": 14953,
                    "screenviewOffset": 14953,
                    "count": 5,
                    "fromWeb": true,
                    "target": {
                        "id": "qty",
                        "idType": -1,
                        "name": "qty",
                        "tlType": "textBox",
                        "type": "INPUT",
                        "subType": "text",
                        "position": {
                            "width": 36,
                            "height": 21,
                            "relXY": "0.6,0.6"
                        },
                        "currState": {
                            "value": "myLoginID"
                        },
                        "isParentLink": false,
                        "prevState": {
                            "value": ""
                        },
                        "dwell": 1609,
                        "visitedCount": 1
                    },
```

Since this value is stored as the value of a JSON path, you must specify the path that uses a different method of identifying its location in the JSON data.

Client Framework	What you do
UI Capture	For AJAX, data blocking is configured in a similar manner to data masking. For UI Capture j2 data blocking is configured by specifying identifiers, the identifier type, and the masking properties through the Configuration Wizard. If a live connection to the web server that hosts the Tealeaf target page is not available, captured UI data is discarded.
Android Logging	The configuration is similar to configuring fields for data blocking.
iOS Logging Framework	The configuration is similar to configuring fields for data blocking.

Privacy configuration for UI Capture

IBM Tealeaf UI Capture enables the blocking and masking of sensitive information within the client browser, before the data is forwarded to IBM Tealeaf for capture, while it allows the data to be forwarded to your web servers for normal processing. Sensitive data that was cleansed through UI Capture never reaches IBM Tealeaf, which ensures that your customer's interactions are secure in UI Capture.

- UI Capture enables the blocking of user input data by element ID, name, or xpath.
- Masks can be expressed as explicit strings, replacements for character types, or custom functions.
- For more information about how data privacy is managed throughout the IBM Tealeaf system, see "Managing Data Privacy in Tealeaf CX" in the *IBM Tealeaf CX Installation Manual*.

  Note: If you have questions about implementing data privacy in UI Capture, contact IBM Tealeaf Professional Services.

To specify a privacy rule, you must define:

- The type of identifier.
- The targets to which the rule applies.
- The type of masking to apply to the targets.

Specifying a privacy rule

In the configuration, a single privacy rule is specified within the `privacy` object by using the following configuration template.

```
{
     targets: [
         {
             id: "htmlid",
             idType: -1
         }
     ],
     maskType: 3
 }
```

Specifying targets

To specify a target, you must specify the following properties.

Note: You can specify multiple `id` or `idType` targets for each masking rule.

Property
Description

`id`
The identifier for the target element. This value is specified according to the `idType` value.
In the configuration file, you can use a regular expression to specify matching identifiers. For example, the following target configuration matches all HTML identifiers that end with `_pii`:

```
message: {
    privacy: [
        {
            targets: [
                {
                    id: { regex: ".+_pii$" },
                    idType: -1
                },
            ],
            "maskType": 3
        }
    ]
}
```

idType
The type of identifier. The following types are supported:

Note: Values for idType are recorded as negative numbers.

- -1 - HTML ID
- -2 - xpath identifier
- -3 - HTML name or other element attribute identifier

CSS selector
In the configuration file, you can also specify CSS selector values to match CSS elements for privacy masking. In the example below, the designated privacy rule is applied to all password input fields:

```
message: {
    privacy: [
        {
            targets: [
                "input[type=password]"
            ],
            "maskType": 3
        }
    ]
}
```

Specifying mask type

IBM Tealeaf UI Capture supports the following mask types (maskType values):

Table 1. Privacy configuration for UI Capture j2

Value	Description	Example	Masked Example
1	Value is blocked and replaced by an empty string.	"HelloWorld123"	""
2	Value is masked with a fixed string of X's	"HelloWorld123"	XXXXX
3	Value is masked according to the following parameters: • a lowercase letter is replaced by x. • an uppercase letter is replaced by X. • a numeral is replaced by 9. • a non-alphanumeric value is replaced by @.	"HelloWorld123"	"XxxxxXxxxx999"
4	Custom function **Note:** A masking function must be defined as maskFunction.	"HelloWorld123	Depends on the defined function

Privacy configuration for Android Logging Framework

In the Android Logging Framework, the settings to control data masking and blocking and the fields to use to specify what is controlled are available in the TLFConfigurableItems.properties file.

The same method of blocking or masking is applied to all items configured to be controlled. You cannot specify multiple methods of blocking or masking.

Default Privacy Configuration for Android

In the default `TLFConfigurableItems.properties` file, the privacy configuration settings are specified to mask data:

```
#Masking settings
HasMasking=true
MaskIdList=com.tealeaf.sp:id\/EditText*,com.tealeaf.sp:id\/login.password
HasCustomMask=true
SensitiveSmallCaseAlphabet=x
SensitiveCapitalCaseAlphabet=X
SensitiveSymbol=#
SensitiveNumber=9
```

In the configuration, privacy in Android is defined as follows:

- Since `HasMasking=true`, privacy is enabled.
- Since `HasCustomMask=true`, a custom mask is applied. So, data masking is enabled. If it was false, then it would use blocking.
- The masking characters are defined in the `Sensitive` settings.

The list of fields in the response data to which to apply the mask is defined in the `MaskIdList`, where fields are delineated by a comma. In the default configuration, there are two fields, defined by using regular expressions.

Field
Description

`com.tealeaf.sp:id\/EditText*`
For the specified namespace, privacy masking is applied to all fields whose `id` includes `/EditText`. For Android applications, this configuration applies privacy to all fields where text is entered, which is the safest, most conservative privacy configuration.

`com.tealeaf.sp:id\/login.password`
For the specified namespace, privacy masking is applied any field that includes `/login.password`, which might correspond to the identifier for the password field in your application.

In the above, the value before the colon in each regular expression (`com.tealeaf.sp`) identifies the namespace to which the regular expression is applied.

Note: Since the default configuration does not specify the namespace of your Android mobile application, privacy is disabled by default for applications that are monitored by the Android Logging Framework.

You can use these configuration settings or modify them to meet the requirements for your application. The following sections describe data blocking and data masking in general, and examples are provided later in the section.

Configuring data blocking

To configure data blocking in the Android Logging Framework, set the following values in the `TLFConfigurableItems.properties` file:

Item ID
Value

`HasMasking`
Set this value to `true`.

`MaskIdList`
Comma-delimited ids or regular expressions to find ids.

`HasCustomMask`
Set this value to `false`.

`SensitiveSmallCaseAlphabet`
Do not specify a value.

SensitiveCapitalCaseAlphabet
Do not specify a value.

SensitiveSymbol
Do not specify a value.

SensitiveNumber
Do not specify a value.

When the HasCustomMask setting is set to false, the masking function returns an empty string, which is inserted in place of the value to be masked.

For more information about these settings, see "Tealeaf Android Logging Framework Configuration File" in the *IBM Tealeaf Android Logging Framework Reference Guide*.

Configuring masking

To configure data masking, you must set HasCustomMask to true and augment the example configuration with the masking characters. These values are set in the TLFConfigurableItems.properties file:

Item ID
Description

HasMasking
Set this value to true.

MaskIdList
Comma-delimited ids or regular expressions to find ids.

HasCustomMask
Set this value to false.

SensitiveSmallCaseAlphabet
This single value specifies the masking character that is applied to lowercase letters. It can be any string value.

SensitiveCapitalCaseAlphabet
This single value specifies the masking character that is applied to uppercase letters. It can be any string value.

SensitiveSymbol
This single value specifies the masking character that is applied to symbol characters. It can be any string value.

SensitiveNumber
This single value specifies the masking character that is applied to numerals. It can be any string value.

Privacy configuration for iOS Logging Framework

Masking functionality is configured with the Masking property in TealeafBasicConfig.plist inside TLFResources.bundle.

Applicable iOS controls

These iOS controls can be masked:

- UITextField
- UITextFieldSecure
- UITextView
- UITextViewSecure
- UIButton
- UILabel
- UISegmentedControl

- `UIAlertView`
- `UIAlertControl`

Masking Levels

A masking level tells the framework how much of the content to preserve. There are two types of masking levels: standard and custom.

The standard masking level replaces the original content with an empty string.

The custom masking level replaces the original content with different characters.

Table 2. How logging levels work

Log Level	Description	Text Entered by User	Text After Privacy
Standard	Data is blocked.	"Password123"	""
Custom	Letters, numbers, and symbols are masked by using different characters.	"Password123"	"Xxxxxxxx#999"

How to enable masking

Masking functionality can be configured with the `Masking` property of `TealeafBasicConfig.plist` file inside `TLFResources.bundle`.

To set the masking level, set the corresponding values for `HasMasking` and `HasCustomMask` properties of the `Masking` list.

Table 3. Setting masking levels

Masking Level	`HasMasking` Property	`HasCustomMask` Property
No Masking	NO	
Standard Masking	YES	NO
Custom Masking	YES	YES

To mask a control, add the regular expression under the `MaskingIdList` property under the `Masking` property to match the ID of the control. All the controls whose IDs match with the regular expression mask based on the configured masking level.

Example: The regular expression `^9[0-9][0-9][0-9]$` matches all the controls whose IDs have four characters, start with 9 and the remaining characters range from 0-9.

Custom masking

You can configure custom masking through the `Sensitive` dictionary in the `TealeafBasicConfig.plist` file.

Table 4. Sensitive dictionary configuration

Configuration	Description
capitalCaseAlphabet	All uppercase letters are masked with the string value for the capitalCaseAlphabet key. In the example, the value is set to X. It is applied for logging levels 2 and 3 in the example.

Table 4. Sensitive dictionary configuration (continued)	
Configuration	**Description**
smallCaseAlphabet	All lowercase letters are masked with the string value for the smallCaseAlphanet key. In the example, this value is set to x.
Number	All numerals are masked with the string value for the number key. In this example, this value is set to 9.
Symbol	All non-alphanumeric characters, such as _ or @, are masked with the string value of the symbol key. In the example, this value is set to #.

Configuring data blocking

To apply data masking, set the standard masking level.

When the configuration is applied to the controls, no values are captured and transmitted to IBM Tealeaf. No other configuration is required.

Migrating old masking functionality to new masking functionality

- Previous versions of the SDK support four levels of masking, which is no longer the case with the current SDK version.
- The current version of the SDK supports only two levels of masking: standard and custom levels.
- The `TagRegex` property of `TealeafBasicConfig.plist`, which takes comma-separated regular expressions, is replaced with the `MaskIdList` array property that takes regular expressions as elements.
- The masking levels that used to be set on individual elements and controls (for example, adding key `UITextField` with the value 3 under the `Masking` property would set `UITextField` with a masking level of 3) is replaced with universal level masking across the application through the properties `HasMasking` and `HasCustomMask`.

Data privacy in UI Capture

IBM Tealeaf CX UI Capture for AJAX enables the blocking and masking of sensitive information within the client browser, before the data is forwarded to Tealeaf for capture, while allowing the data to be forwarded to your web servers for normal processing.

Blocked data

Sensitive data that is cleansed through UI Capture never reaches Tealeaf, which ensures that your customer's interactions are secure in UI Capture.

- UI Capture enables the blocking of user input data by element ID or name or both.
- Masks can be expressed as explicit strings, replacements for character types, or custom functions.

Specifying for JSON messages

Beginning in Release 8.5, the IBM Tealeaf CX platform supports the capture of JSON-based messages from Tealeaf client frameworks, including a version of IBM Tealeaf UI Capture.

- Each JSON message corresponds to an individual step in the visitor's session. Multiple steps may exist on a single hit, and you can create events for individual steps of a hit. See "Step-Based Eventing" in the *IBM Tealeaf Event Manager Manual*.

Block array fields and mask types

The data and masking functions to apply privacy are defined as objects in the `tlFieldBlock` array that can be found in `TealeafSDKConfig.js` file of the UI Capture package. For each masking operation applied through UI Capture, you must specify a different row in the `tlFieldBlock` array.

Field block array fields

For each masking operation applied through UI Capture, you must specify a different row in the `tlFieldBlock` array. This array contains these fields:

Field name	Description
`name`	A regular expression string that specifies the name of the element on the page to which to apply privacy. You can specify names, IDs, or both within a single record.
`id`	A regular expression string that specifies the ID of the element on the page to which to apply privacy. You can specify names, IDs, or both within a single record.
`caseinsensitive`	When set to `true`, UI Capture attempts a case-insensitive match the name or ID. For example, `MyAttribute` and `myattribute` both match.

Mask types

Tealeaf provides three methods for masking or blocking sensitive data.

Mask Type	Description	JavaScript Function
Preserve Mask	Use this mask type to specify the mask to use for alphanumeric characters. Mask is specified in the `tlPrivacyMask` array.	`TeaLeaf.Client.PreserveMask(element);`
Basic Mask	Use this mask type to mask data with a pre-defined string, such as XXXXXX	`TeaLeaf.Client.BasicMask(element);`
Empty Mask	An empty mask removes all characters and does not replace them.	`TeaLeaf.Client.EmptyMask(element);`
Custom Mask	You can replace any of the other calls to Tealeaf masking functions with a call to your own function.	N/A

These methods are available as JavaScript functions, which are referenced in the declarations of each field in the `tlFieldBlock` array.

Privacy mask

If needed, you can specify a privacy mask such that a different masking character is applied depending on the type of character. The `tlPrivacyMask` is specified by using these fields:

Field	Description
`upperChar`	This character is applied to uppercase alphabetical characters. The default value is X.

Field	Description
lowerChar	This character is applied to lowercase alphabetical characters. The default value is x.
numericChar	This character is applied to numeric characters. The default value is 9.
symbolChar	This character is applied to symbol characters. The default value is #.

Field Masking
You can use UI Capture to add a mask over sensitive data that must be validated on the server. In the following example, credit card-related fields are masked by dummy characters to indicate that the data entered. After the changes are made to TealeafClientCfg.js, the dummy field values are submitted to the server and processed by Tealeaf.

Mask fields by using names and IDs

You can specify names and IDs by listing explicit values or by using regular expressions. Based on the evaluation of these expressions, the specified privacy operation can be applied to one or more page elements, including all page elements.

In the tlFieldBlock data array, you reference fields by their unique identifiers (IDs). If IDs are unavailable, you can reference fields by name, which might not be unique.

To apply a privacy operation to a single named field:

```
{"name": "creditcard", ...
```

To apply a privacy operation to multiple named fields, separate the fields by the pipe character (|):

```
{"name": "creditcard|password", ...
```

You can also apply regular expressions to match names or IDs based on patterns. Suppose that you defined page identifiers that require UI Capture privacy to have pvt prefix in their ID value. To apply privacy, use the following regular expression as the value for the ID:

```
{"id": "^pvt", ...
```

You can also specify the same privacy masking operation for names and identifiers in a single record. In this example, two regular expressions are specified for matching names and IDs. These expressions match all values for the name and ID attributes, effectively blocking all element values from the page.

The result is that everything that is monitored by UI Capture is masked or blocked, based on the rule.

```
{"id": ".*", "name": ".*", ...
```

Mask fields by using ClassName

You can specify the fields to mask based on the CSS class for the fields. This method allows global masking of any field that is identified by using a common class.

Mask fields by using names and IDs

You can use regular expressions to specify the `classname`. In this example, all fields with CSS classes that have the `priv` prefix are masked.

```
{"classname": "priv.*", ...
```

Specifying values for `name` or `id` is not required.

Function declarations

The masking function API is:

```
string mask(DOMNode element);
```

Where:

- `element` is the DOM node with the matching name or ID attribute as specified in the masking record.
- `string` is the masked return value that is sent by Tealeaf in the UI POST.

Limitations

UI Capture can only mask or block data that is collected through the IBM Tealeaf UI Capture. This library does not provide access to data contained on the page, which is not managed by the library. The following examples cannot be made private by UI Capture:

- A visitor ID embedded in the HTML of the page
- A static element that is not captured by the library, which contains the visitor's account balance or Social Security number, or similar.

These types of sensitive data must be masked or blocked after they reach Tealeaf.

Performance

Care must be taken to keep the masking records to a minimum. Only use optimized regular expressions and optimized code in your custom functions. A poorly configured privacy mask can have adverse impacts on the performance of the page.

Example of blocking data from transmission

You can remove the data from being submitted to the server.

This example blocks data by referencing the `EmptyMask` function in the declaration:

```
{"id": "myElement", "caseinsensitive": false, "mask":
  function () { return TeaLeaf.Client.EmptyMask.apply(this, arguments); }}
```

Default Configuration

This is the default specification for `tlFieldBlock` that is provided by Tealeaf.

The default configuration does not perform any masking. You must modify this code to support your privacy requirements before you deploy the IBM Tealeaf UI Capture.

```
tlFieldBlock:[
        /* Sample block rules:
          // Mask all field names that have "creditcard" or "password"
          // substrings using the PreserveMask() function.
          {"name": "creditcard|password", "caseinsensitive": true,  "mask":
            function ()
              { return TeaLeaf.Client.PreserveMask.apply(this, arguments); }}
```

```
        // Mask all field ids that match pvt0, pvt1 ... pvt9 using
        // the EmptyMask() function.
        {"id": "^pvt[0-9]$",           "caseinsensitive": true,  "mask":
          function ()
            { return TeaLeaf.Client.EmptyMask.apply(this, arguments); }}
        // Paranoid mode: Mask all name and id values with the
        // BasicMask() function.
        {"id": ".*", "name": ".*",     "caseinsensitive": false, "mask":
          function ()
            { return TeaLeaf.Client.BasicMask.apply(this, arguments); }}
      */
      ],

        // The mask used by the PreserveMask() masking function.
        tlPrivacyMask: {
          "upperChar":   "X",
          "lowerChar":   "x",
          "numericChar": "9",
          "symbolChar":  "#"
        },
```

Tealeaf configuration for client frameworks

You do several tasks to configure Tealeaf for client frameworks. Some of the tasks are done on the client frameworks and some on the server.

Process

When you configure Tealeaf and the client frameworks, you:

1. Install and implement the client framework, Android, iOS, or UI Capture
2. Configure session agents
3. Configure PCA

Client framework tasks

Use the client framework documentation to configure Tealeaf for client frameworks:

1. Deploy and configure the client framework.
2. Set up data privacy in the client framework.
3. Configure sessionization on the client framework.
4. Configure sessionization on the server.
5. Configure the target page.

Windows pipeline, Reference and session agent, and WURFL

When data is captured and passed through Tealeaf, session information is analyzed and sometimes transformed in the Windows pipeline. The pipeline is a series of configurable session agents. Each of the agents performs one or more operations on the hit data before the agent passes the hit to the next agent in the sequence.

Whether a mobile native application submits a user agent string is determined by the developer of the application. To facilitate user agent detection, the Tealeaf client frameworks for Mobile App automatically submit a user agent string and other device properties, which the Tealeaf Reference session agent uses to populate the same values in the request that are used by Mobile web.

The Tealeaf Reference session agent is deployed in the default pipeline to verify user agent information that is submitted by the client with each request. When a web browser or fixed browser submits a request to a server, it typically includes a string to identify itself. This string is used by the Tealeaf Reference session agent to perform lookups against the WURFL public standard, which identifies mobile user agents. When WURFL is downloaded from its source and deployed, more useful information about the

browser, operating system, and platform is extracted based on the user agent and inserted into the request of the hit.

Session agent configuration

Windows pipeline session agent configuration varies based on how data is submitted and captured. Most of the frameworks do not require additional configuration. This table lists the frameworks and whether additional configuration is needed:

Framework	Configuration
Mobile native applications	For mobile native applications that use the step-based method of submitting and capturing data, no additional configuration is required in the Windows pipeline. For mobile native applications that use the legacy, hit-splitting method of processing data inside of Tealeaf, some session agents must be deployed in your Windows pipeline.
UI Capture	If you are using the JSON version of IBM Tealeaf UI Capture, no additional configuration in the Windows pipeline is required. For mobile web applications captured by UI Capture, deploy the Tealeaf Reference session agent to capture user agent information for mobile web browsers. The Tealeaf Reference session agent is included by default in the Windows pipeline.

PCA Configuration

By default, the IBM Tealeaf CX Passive Capture Application is configured to enable the capture of the JSON mimetypes that is submitted by the Tealeaf client frameworks. Before you continue, verify that these types are enabled for capture.

Each Tealeaf Logging Framework might be using a different content type for submitting events for capture to Tealeaf. Be sure to review and verify the content type for each deployed client framework.

Framework	Required Content Type to Capture
IBM Tealeaf UI Capture	`application/json`
Android Logging Framework	`text/json`
iOS Logging Framework	`text/json`

Next steps

After the PCA was configured to capture client framework data and the client framework was properly deployed, data is being captured by IBM Tealeaf.

After all configuration is complete, you can test for the presence of client framework data and define IBM Tealeaf events and event-related objects to enable search and replay of sessions and to provide insightful analysis of your applications.

Integrate client framework data into Tealeaf

The Tealeaf client frameworks let you capture user interface events and application events from the visitor's client, which might not generate a full request from the web application. These client frameworks are Javascript-based toolkits that are deployed through your web application and installed in the visitor's client.

Without interrupting the customer experience, the client frameworks monitor the client application for events of interest and transmit those events to Tealeaf for capture and processing. Through the client frameworks, you can apply the same level of monitoring on your mobile or web applications as you can through the browser-based applications that are monitored by IBM Tealeaf cxImpact.

Before you begin, one or more of the Tealeaf clients frameworks must be licensed, deployed, and configured in your web application environment. Tealeaf client frameworks require the IBM Tealeaf CX Mobile license.

General workflow

The general workflow for integrating client framework data into Tealeaf is:

1. Locate client framework sessions, including differences in how the data is posted.
2. Identify user agent information that is provided by the client framework.
3. Create step attributes from them to gather user agent data from the logging framework(s).
4. Create events from the step attributes.
5. Integrate events into existing reporting.

Prerequisites

Before you integrate client framework data into Tealeaf you must install software, make sure that your accounts have access to the TealeafTealeaf components, configure one of the client frameworks, and enable and configure parsers.

List of prerequisites

Before you integrate client framework data into Tealeaf you must:

1. Install IBM Tealeaf CX Release 8.4.1 or later.
2. Verify that your Portal account has access to the:
   - Tealeaf Event Manager. To access the Tealeaf Event Manager, select **Configure** > **Event Manager** from the **Portal** menu.
   - Tealeaf Report Builder. To access the Tealeaf Report Builder, select **Analyze** > **Report Builder** from the **Portal** menu.
3. Integrate one of the client frameworks is integrated with your Tealeaf application.
4. Verify that extended user agent parsing is enabled. Extended user agent parsing is enabled by default through the Tealeaf Reference session agent, which is required in most pipelines. Use TMS to verify that extended user agent parsing is enabled.
5. Configure user agent parsing, if necessary.
6. For UI capture only, if you deployed a version of your web application for mobile web browsers, deploy the WURFL public standard to enable detection of mobile-based user agents.
7. For UI capture only, after you enabled user agent detection, configure events to detect mobile device characteristics that are based on the captured and verified user agent information.

Required licenses

Tealeaf client frameworks can be deployed to capture data from the types of applications that are listed and submitted to Tealeaf for processing. Before you begin, it is assumed that you installed and

successfully deployed one of the Tealeaf client frameworks and that you verified client framework data is being captured by Tealeaf.

Framework	Required license
UI Capture j2	• CX-Extended platform license
Tealeaf Android Logging Framework	• CX-Extended platform license • Tealeaf CX Mobile license
Tealeaf iOS Logging Framework	• CX-Extended platform license • Tealeaf CX Mobile license

Use groups and labels to store events and event-related objects

Before you begin building events and event-related objects, you should consider creating groups and labels to store them. You may build hit attributes, events, session attributes, and dimensions, so you should try to find ways to create consistent labels in each of the appropriate tabs in the Event Manager.

This table shows lists and describes one set of labels and groups for client framework data:

Label/Group	Description
`Source - UIC`	Event objects created to track data from Tealeaf IBM Tealeaf UI Capture
`Source - Android`	Event objects created to track data from the IBM Tealeaf Android SDK
`Source - iOS`	Event objects created to track data from the IBM Tealeaf iOS SDK
`Source - Mobile`	Event objects created to track data from mobile devices in general

Locating client framework sessions

This section describes the identifying markers in client framework sessions and the objects that are provided by Tealeaf to locate these markers.

Identify client framework sessions

When client framework messages are received by the PCA, they are examined for the source of the message. Based on the source that is detected in the `x-tealeaf` header in the raw request, the PCA inserts header values for each client framework as a name-value pair in the `[env]` section of the request:

Client Framework	Name/Value Example
IBM Tealeaf UI Capture (JSON)	`HTTP_X_TEALEAF=device (uic) Lib/2012.06.01.1.JS` This header value is inserted only if UI Capture is submitting JSON-based data.

Client Framework	Name/Value Example
IBM Tealeaf UI Capture (XML)	`HTTP_X_TEALEAF=ClientEvent` For IBM Tealeaf UI Capture solutions that use XML-based messaging, you can locate sessions by searching for this data in the request or for events that you create to identify this exact name-value pair in the request.
Android Logging Framework	`HTTP_X_TEALEAF=device (android) Lib/0.0.10`
iOS Logging Framework	`HTTP_X_TEALEAF=device (iOS) Lib/8.5.4.1`

In the above name-value pairs, the value after `Lib/` indicates the version number of the client framework that captured and submitted the hit.

Note: Tealeaf provides some event objects for tracking the above items. These items are described below. If you want to create extra step attributes and events from the HTTP_X_TEALEAF request variable, you must create them manually through the Event Manager. When these objects are created through the recommended method in BBR, the Event Manager identifies them as existing in the provided set of event objects. You can work around this safeguard by manually creating them.

Objects provided by Tealeaf for mobile native applications
For mobile native applications, Tealeaf provides a set of objects to help integrating the data into your Tealeaf data set.

Mobile Device objects

This table lists and describes the objects provided for mobile apps:

Sequence	Object	Description
1	`Mobile Device Type` hit attribute	The `Mobile Device Type` hit attribute that is provided by Tealeaf is designed to identify hits that was submitted from the mobile client frameworks. **Note:** This hit attribute filters out values that were captured from the IBM Tealeaf UI Capture solution that were posted in the HTTP_X_TEALEAF header. Values that are reported in this hit attribute are for mobile native applications only. • A **hit attribute** is a Tealeaf event-related object that checks for patterns of text or specific strings in each request that passes through the event engine. • Hit attributes are defined in the Event Manager, which is available through the Tealeaf Portal.

Sequence	Object	Description
2	Mobile Device event	This event that is provided by Tealeaf is triggered if the Mobile Device Type hit attribute is found. The recorded value is the value of the hit attribute, which is the value between the parentheses of the request variable.
3	Mobile Device dimension	This dimension is recorded from the last value in the session for the Mobile Device event. Dimensions are defined in the Event Manager, which is available through the Tealeaf Portal.
4	Traffic Type dimension	When the value for the HTTP_X_TEALEAF variable includes iOS or android, the value for the Traffic Type dimension is set to MOBILE_APP. In this manner, you can segment reporting based on whether the type of traffic is sourced from a mobile native application.

Sample values to create hit attributes for mobile web browsers
Sessions that are sourced from mobile web browsers are captured by Tealeaf with the IBM Tealeaf UI Capture client framework. Tealeaf provides objects to detect and capture data from mobile web browser sessions.

Values for a hit attribute to detect sessions sourced from UI Capture

To detect mobile native sessions that were captured from IBM Tealeaf UI Capture, you need to detect for the UIC value in the HTTP_X_TEALEAF request variable.

Below, you can see the values that you need to assign for the Start Tag and End Tag for this hit attribute:

- Start Tag:

```
\r\nHTTP_X_TEALEAF=device (UIC)
```

- End Tag: Leave this blank.

This hit attribute applies to Release 8.5 or later. If you are using the legacy XML version of IBM Tealeaf UI Capture, the value of the hit attribute (after the equals sign) must be ClientEvent exactly.

This condition identifies that the hit was sourced from IBM Tealeaf UI Capture.

You should create an event specifically to detect for the presence of this value in the CUI Hit attribute. If this value changes in the future, you can update all of your event configurations by updating the detecting event.

Values for a Mobile Web Hit event

IBM Tealeaf UI Capture can be used to track client activities for fixed browser and mobile web applications. Mobile web applications are applications that are served to mobile clients that are experienced through a web browser on the client device. You might find it useful for reporting purposes to create a hit attribute and event to detect IBM Tealeaf UI Capture sessions that were served from a mobile client browser.

When you create your event, you must add a condition that identifies the hit as being sourced from a recognized mobile device and submitted through IBM Tealeaf UI Capture.

The `Traffic Type` attribute is used to detect the type of traffic, whether it is sourced from a fixed browser, mobile browser, or some form of bot traffic. When the value of this attribute is `MOBILE`, the hit is sourced from a mobile web browser, which also means that the session was captured by IBM Tealeaf UI Capture.

When the attribute is added, your conditions for your event should look like:

```
Traffic type:  First Value  Equals MOBILE
```

The value of the hit attribute must be `MOBILE` exactly. Case-sensitive matching is not required.

The event must be configured to be triggered on a hit trigger (not a session trigger) and all event conditions must be met for the event to fire.

This event is now configured to fire if the hit was received from IBM Tealeaf UI Capture from a mobile web browser session.

Use this event as a condition to create other events, making sure that in most cases, for proper function of the event, the hit conditions are configured so that all of the conditions are met.

Sample values to use to create hit attributes for any client framework
Hit attributes are defined in the Event Manager, which is available through the Tealeaf Portal.

CUI Hit attribute

The `CUI Hit` hit attribute checks for the presence of the `HTTP_X_TEALEAF` request variable and, if found, gathers the value. This request variable indicates that it was submitted from one of the client frameworks. The `CUI Hit` has these characteristics:

1. Name: CUI Hit
2. Description: Hit count, as reported from client user interface. Requires Tealeaf UI Capture.
3. Active: Selected
4. Search in: Request
5. Use: Start Tag/End Tag
6. Start Tag:

   ```
   \r\n\HTTP_X_TEALEAF=
   ```

7. End Tag:

   ```
   \r\n\
   ```

8. Case sensitive: Selected
9. Encoding: UTF-8
10. Change Case: No change

If the `CUI Hit` request variable is present, then the value of the request variable is stored as the hit attribute value.

The `CUI Hit` hit attribute is configured to:

- Search in the request
- Search using a start tag and an end tag.
  - Start Tag:

    ```
    \r\n\HTTP_X_TEALEAF=
    ```

  - End Tag:

```
\r\n\
```

In the Start and End tags, the string `\r\n` is used to reference a return/newline that signals the ending of one line and the beginning of a new one in captured data. When the request is scanned and the `\r \nHTTP_X_TEALEAF=` string is detected, all values between that string and the end tag string (`\r\n`, which indicates the end of the line) are gathered as the value for the hit attribute `CUI Hit`.

Whenever `CUI Hit` is used for other event objects evaluated on the same hit, the value captured by the hit attribute is value that is used.

When the next hit arrives in the Event Engine, the hit attribute is retested, and potentially a new value is stored in `CUI Hit` for reference.

Differences from Mobile Device hit attribute

You might notice that the `Mobile Device` hit attribute is also scanning the HTTP_X_TEALEAF request variable.

- The `Mobile Device` hit attribute is interested in only the values between the two parentheses in the request value.
- The `CUI Hit` hit attribute gathers all values until the end of the line. Therefore, it gathers a much wider data set and can be used as a test of whether the hit was sourced from any Tealeaf client framework.

Attributes and events used to detect sessions

After you create event objects, you can see how sessions sourced from each framework is detected within Tealeaf. Tealeaf uses the combination of hit attributes, events, and dimensions to detect the source of the session captures.

Source of Session Capture	Hit Attribute	Event	Dimension	Session Attribute
Android LF	Mobile Device Type	Mobile Device	Mobile Device	none
iOS LF	Mobile Device Type	Mobile Device	Mobile Device	none
UIC for JSON	Create a hit attribute to detect sessions sourced from UI capture.	none	none	none
UIC for XML	CUI Hit (value must be ClientEvent)	none	none	none

Sample values to create events to detect messages from mobile native application sessions

Based on the event objects provided and that you created, you can detect for sessions that were sourced from one of the mobile client frameworks or from a specific client framework

Example event to detect messages from any mobile client framework

If you are creating events to monitor data from any mobile client framework, use the `Mobile Device Type` hit attribute as an event condition. Set the operator to be `Hit Attribute Found`

In this example, the event conditions are defined to detect for the presence of the `Mobile Device Type` and a user-defined attribute, `Mobile Alert Message`. When both attributes exist, the event fires when a mobile alert message is sourced from one of the logging frameworks for mobile devices.

Example event to detect a specific mobile client framework

You can also create events to track activities from a specific client framework.

For example, suppose you want to monitor mobile alert messages from your iOS application, independently of those of a deployed Android application. In this case, you define your event conditions to detect for the presence of a specific value for the `Mobile Device Type`:

You may want to create an event that detects for the specific value for the `Mobile Device Type` hit attribute and records that value. You might name that event something like `Mobile Session - ios`.

Sample values used to detect mobile web sessions
Some sessions that are initiated by mobile devices might not be captured by the android or iOS frameworks. Depending on your web application, it might be possible for visitors to interact with your web application using a mobile web browser. Sessions that are initiated in this manner are not detected and processed through the Logging Frameworks.

These types of interactions apply to IBM Tealeaf CX Mobile for Mobile Web.

Capture and detection of sessions that are sourced from mobile web browsers requires UI Capture.

Sessions from mobile web browsers that are captured from the IBM Tealeaf UI Capture solution are detected by extended user agent parsing, which generates the data to report them as `MOBILE` sessions in the `Traffic Type` hit attribute, which is the source for the `Traffic Type` dimension.

This event might already be created.

Tracking type of session
There are three sample events to identify sessions that originate from a Tealeaf client framework-enabled application. Two are provided by Tealeaf and one you create.

this table summarizes events that you can use to eidentify sessions that originate from a Tealeaf client framework-enabled application:

Table 5. Tracking type of session

client framework	submitted data type	HTTP_X_ TEALEAF value	Detected value	Capturing event
UIC for XML	XML	`ClientEvent`	`ClientEvent`	`CUI Hit`
UIC for JSON	JSON	`device (UIC) Lib/2012.06. 01.1.JS`	`UIC`	You must create an event.
Android	JSON	`device (Android) Lib/0.0.10`	`Android`	`Mobile Device`
iOS	JSON	`device (iOS) Lib/8.5.4.1`	`iOS`	`Mobile Device`

For Release 8.4, the mobile logging frameworks used the `HTTP_X_TEALEAF_DEVICE` request variable, which populated the `Mobile Hit` hit attribute. For Release 8.5 and later, this attribute is no longer used.

Creating a session attribute to track session type across events
To track the type of session across all of your events, create a session attribute.

About this task

Call the attribute `CUI Session Type`.

Procedure

1. In the Event Manager, click the **Session Attributes** tab.
2. Click **New Session Attribute...**.
3. Key Properties:

   Property
   Description

Name
Use `CUI Session Type` or similar.

Description
Enter something similar to the following: `Source of session if it is captured from the client using a Tealeaf client framework`.

Populated By
Select one of the events that you created.

4. Click **Save Draft**.

Results

The session attribute is saved.

Edit events to update the session tracking attribute
Modify the events that you want to update the session tracking attribute. If you want mobile session events to update this attrbitute,

Procedure

1. Click the **Events** tab.
2. Right-click one of the events you created and select **Edit Event...**.
3. Add the attribute you created to the lists of attributes that this event updates:
   a) Click the **More Options** tab.
   b) Next, to Update Session Attribute, click **Select?**.
   c) Select the session attribute that you just created.
   d) Click **Save Draft**.
4. Repeat these steps for the other events you created, for example the mobile .
5. To save all changes, click **Save Changes**.

Results

When the created events are triggered, they record the value of the event, which is the identifier for the type of CUI session, to the session attribute.

Adding a dimension to be sourced from session attribute
Whenever an event is triggered, the session attribute is updated, and any dimensions associated with the event are updated. A **dimension** is a piece of contextual information that is recorded when an event fires. You can use dimensions to create reporting filters to segment reporting along the type of client framework session.

About this task

Because the events are populated only with values that are defined by the Tealeaf client frameworks, you create the whitelist to capture these values. These values should be the same as those recorded for the HTTP_X_TEALEAF request variable.

Procedure

1. Click the **Dimensions** tab.
2. Click **New Dimension**.
3. For the Populated By option, select the session attribute you created.
4. Click the **Advanced Options** caret.
5. For Values to Record, select `Whitelist Only`.
6. Click **Edit Whitelist?**

7. Specify the values in the whitelist.
8. Save the whitelist.
9. Populate the other dimension properties as needed.
10. Save the dimension as a draft.
11. Save changes to the server.

Results

This dimension can be associated with other events to segment reporting along the type of client framework session.

User Agent Detection

You can configure Tealeaf to detect and capture user agent information that is submitted from the client application.

User agent standards

This table lists and describes the user agent standards that Tealeaf uses. Standards are listed by application type:

Application type	User agent standard	Description
desktop browser	browscap	Tealeaf uses the browscap standard for capturing user agent information that is submitted from desktop browsers
mobile desktop browser	WURFL	IBM Tealeaf CX Mobile can use the WURFL public standard for capturing user agent information that is submitted from mobile web browsers
mobile native application	n/a	User agent information is submitted in a consistent form by the client framework monitoring your mobile native applications.

One of the functions of the Tealeaf Reference session agent is to perform lookups against these public standards for user agent information that is submitted from the client.

The Tealeaf Reference session agent must be included in every Windows pipeline in which session data is processed.

User agent detection for UI Capture

Since the IBM Tealeaf UI Capture solution is provided for applications being served to fixed desktop browsers or mobile web browsers, user agent information is available as part of regularly submitted requests from these browsers, using Browscap or WURFL.

Tealeaf reference dimensions from IBM Tealeaf UI Capture solutions is written to different values in the request.

User detection and mobile native applications

Typically, individually mobile applications send a unique user agent string that is not known to any public repository, or they may not send a user agent at all. As a result, user agent detection for mobile native applications cannot rely on WURFL, Browscap, or other public standard.

HTTP_X_TEALEAF header and mobile native applications

To enable tracking of user agent-related information for mobile native applications, the Tealeaf client frameworks submit the HTTP_X_TEALEAF header, which is rendered into these request variables:

Logging Framework	Example submitted header
Android Logging	`HTTP_X_TEALEAF=device (android) Lib/0.0.10`
iOS Logging Framework	`HTTP_X_TEALEAF=device (iOS) Lib/8.5.4.1`

HTTP_X_TEALEAF_PROPERTY header and mobile native applications

The Tealeaf client frameworks also submit the HTTP_X_TEALEAF_PROPERTY header, which contains user agent information that is extracted from the device itself.

The Tealeaf Reference session agent then uses the properties included in the above header to write out available values in the `[ExtendedUserAgent]` section, in the request variables that are used by other Tealeaf components to extract user agent information. In this manner, information that provided directly by the Tealeaf client frameworks is used to populate user agent information throughout Tealeaf without requiring a lookup or other reference data.

Searching for client framework sessions

The JSON messages that are submitted from the client frameworks for capture are automatically indexed for search. Each JSON POST is converted to XML and indexed as a **Request/Request** body field.

Example of raw JSON

this is an example of a ras JSON message POST:

```
{"serialNumber":0,"messageVersion":"0.0.0.3","sessions":[{"startTime":133518041
5973,"id":"C6A2913845DA422381FC9678856F6000","messages":[{"context":{"type":"LO
AD","name":"HomeActivity_1335180416289"},"offset":318,"type":2,"contextOffset":
0},{"offset":369,"type":1,"contextOffset":50,"mobileState":{"orientation":0,"fr
eeStorage":32235520,"androidState":{"keyboardState":"HIDDEN_FALSE"},"battery":5
0,"freeMemory":164397056,"connectionType":"UMTS","carrier":"Android","networkRe
achability":"ReachableViaWWAN","ip":"0.0.0.0"}},{"offset":1357,"type":1,"contex
tOffset":1037,"mobileState":{"orientation":0,"freeStorage":32235520,"androidSta
te":{"keyboardState":"HIDDEN_FALSE"},"battery":50,"freeMemory":163835904,"conne
ctionType":"UMTS","carrier":"Android","networkReachability":"ReachableViaWWAN",
"ip":"0.0.0.0"}},{"customEvent":{"name":"Screenshot Taken for file:
android.widget.LinearLayout_1335180417333.png"},"offset":2651,"type":5,"context
Offset":2332}]}],"clientEnvironment":{"mobileEnvironment":{"android":{"keyboard
Type":"QWERTY","brand":"generic","fingerPrint":"generic
\/sdk\/generic:2.3.3\/GR
I34\/101070:eng\/test-keys"},"totalMemory":164151296,"totalStorage":12288,"orie
ntationType":"PORTRAIT","appVersion":"1.0.6","manufacturer":"unknown","userId":
"android-build","locale":"English (United
States)","deviceModel":"sdk","language":"English"},"width":480,"height":800,"os
Version":"2.3.3"}}
```

Example of indexed text

This is an example of the JSON message confverted to XML and indexed:

```
<RequestBody>
 <clientEnvironment>
  <height>800</height>
  <mobileEnvironment>
   <android>
```

```
    <brand>generic</brand>
    <fingerPrint>generic/sdk/generic:
       2.3.3/GRI34/101070:eng/test-
       keys</fingerPrint>
    <keyboardType>QWERTY</keyboardType>
   </android>
   <appVersion>1.0.6</appVersion>
   <deviceModel>sdk</deviceModel>
   <language>English</language>
   <locale>English (United States)</locale>
   <manufacturer>unknown</manufacturer>
   <orientationType>PORTRAIT</orientationType>
   <totalMemory>164151296</totalMemory>
   <totalStorage>12288</totalStorage>
   <userId>android-build</userId>
  </mobileEnvironment>
  <osVersion>2.3.3</osVersion>
  <width>480</width>
 </clientEnvironment>
 <messageVersion>0.0.0.3</messageVersion>
 <serialNumber>0</serialNumber>
  <sessions>
   <id>C6A2913845DA422381FC9678856F6000</id>
    <messages>
     <context>
      <name>HomeActivity_1335180416289</name>
      <type>LOAD</type>
     </context>
     <contextOffset>0</contextOffset>
     <offset>318</offset>
     <type>2</type>
    </messages>
    <messages>
     <contextOffset>50</contextOffset>
     <mobileState>
      <androidState>
       <keyboardState>HIDDEN_FALSE</keyboardState>
      </androidState>
      <battery>50</battery>
      <carrier>Android</carrier>
      <connectionType>UMTS</connectionType>
      <freeMemory>164397056</freeMemory>
      <freeStorage>32235520</freeStorage>
      <ip>0.0.0.0</ip>
      <networkReachability>ReachableViaWWAN</networkReachability>
      <orientation>0</orientation>
     </mobileState>
     <offset>369</offset>
     <type>1</type>
    </messages>
    <messages>
     <contextOffset>1037</contextOffset>
     <mobileState>
      <androidState>
       <keyboardState>HIDDEN_FALSE</keyboardState>
      </androidState>
      <battery>50</battery>
      <carrier>Android</carrier>
      <connectionType>UMTS</connectionType>
      <freeMemory>163835904</freeMemory>
      <freeStorage>32235520</freeStorage>
      <ip>0.0.0.0</ip>
      <networkReachability>ReachableViaWWAN</networkReachability>
      <orientation>0</orientation>
     </mobileState>
     <offset>1357</offset>
     <type>1</type>
    </messages>
    <messages>
     <contextOffset>2332</contextOffset>
     <customEvent>
      <name>Screenshot Taken for file:
         android.widget.LinearLayout_1335180417333.png</name>
     </customEvent>
     <offset>2651</offset>
     <type>5</type>
    </messages>
   <startTime>1.33518e+012</startTime>
  </sessions>
</RequestBody>
```

Searching session using text

Use this task as an example of how to search sessions using text.

About this task

Through the Portal, you can use the following methods to search for these types of information:

Table 6. Searching Using Text

Type	Example	Method of Search
Value of JSON property	`ReachableviaWWAN`	Values are searchable using the `Text in Request` field. See "Searching Session Data" in the *IBM Tealeaf cxImpact User Manual*.
Name of JSON property	`networkReachability`	Use the form field search criterion to specify the name of a JSON property for which you would like to search. See "Searching Session Data" in the *IBM Tealeaf cxImpact User Manual*.

Procedure

Searching sessions using events

In the steps below, you search for an event + dimension value combination. In this case, you can use the `Hit Count` event and one of the following dimensions. Since this event is already associated with these dimensions and is present in all completed sessions, no further configuration is required.

Table 7. Searching Using Events

Type of Search	Dimension	Dimension Value	Description
For specific type of mobile device	`Mobile Device`	`ios` or `android`	The Mobile Device dimension contains the appropriate value if the hit was captured from the named type of device. • See "Searching for mobile native application sessions from a specific type of mobile device" on page 31.
For any type of mobile device	`Traffic Type`	`MOBILE_APP`	The Traffic Type dimension contains the `MOBILE_APP` value if the hit was captured from a mobile native application. • See "Searching for mobile native application sessions from any device type" on page 31.

Searching for mobile native application sessions from a specific type of mobile device

Use this task as an example of how to search completed sessions for a specific type of mobile application session, such as Android or iOS.

About this task

Depending on the volume of traffic to your web application, you might have to wait a while before sufficient sessions are captured and processed by Tealeaf so that you can search for them using this event.

This example shows how to search Completed sessions, but you can also search for active sessions.

You can narrow your search by adding extra search terms. They do not have to be event-based search terms.

Procedure

1. In the **Portal** menu, select **Search** > **Completed Sessions**.
2. For the Search Range, select `Only Today`.
3. Clear the search criteria from the search panel.
4. In the left navigation panel, click **Events**. Do not select the Event Values category.
5. Click the **<select an event>** link.
6. In the Event Selector, select the `Hit Count` event.
7. Click **Select**.
8. The Hit Count event is selected. In the **Search** panel, click **Any Dimension**.
9. Select **Mobile Device**.
10. The Dimension Value Selector is displayed. Select either Android or iOS.
11. Click **Select**.
12. You might want to save this search for future use. To save it, click the **Save** icon in the toolbar. Provide a name for the saved search, and click **Save**.
13. To start the search, click **Search**.

Searching for mobile native application sessions from any device type

About this task

In this section, you learn to search for mobile native applications that were sourced from any device type. Using the `Traffic Type` dimension, you can locate mobile native application sessions.

Procedure

1. Repeat the above steps to select the `Hit Count` event. If you still have the search open, you can continue with the following steps.
2. In the search term panel, click the dimension indicator. In the above example, this link is `Mobile Device - iOS`.
3. In the Dimension Selector, select **Traffic Type**.
4. Click **Select**. In the Dimension Value Selector, select **MOBILE_APP**.
5. The search term is specified, like the following:

6. You might want to save this search for future use. To save it, click the **Save** icon in the toolbar. Provide a name for the saved search, and click **Save**.
7. To start the search, click **Search**.

Note: You can narrow your search by adding more search terms. They do not have to be event-based search terms.

Results

- For more information about session search through the Portal, see "Searching Session Data" in the *IBM Tealeaf cxImpact User Manual*.

Reviewing session search results
You view the results of a session search in a session list. Search results are displayed in a session list.

About this task

When sessions are returned, they are displayed in a session list.

More information about each event is available when you mouse-over the listed event.

As needed, you can configure session lists to contain columns of your choosing.

This procedure has you open a session and search the session request for the

- message header to see if a mobile device was used for the session
- traffic type to see if the traffic was from a mobile device

Procedure

1. In the left column, click the **Replay** icon.
2. Select **BBR**.
3. When the session is loaded, click the topmost entry in the Navigation pane on the left.
4. In the toolbar, click **Request**.
5. The displayed text is the request for the first hit of the session. Select in the request. Press CTRL+A or Command+A to select all of the text.
6. Paste the text into a text editor.
7. Search for:

```
HTTP_X_TEALEAF=
```

8. If the request variable is found, verify that the value listed next to the above string contains either (iOS) or (Android). If so, it is a hit from a mobile native application session.
   - If your Mobile Session event definition includes mobile web sessions that are captured through UI Capture, you might see value of ClientEvent for this request variable.
9. Search the request for:

```
TLT_TRAFFIC_TYPE=
```

   - If the value for the above is MOBILE, it is a hit from a mobile web session.
   - If the value is MOBILE_APP, it is from a hit from a mobile native web application.

Testing events to verify that they work as expected
After you identified a session with your event, you might find it useful to send to the Event Tester. In the Event Tester, you test events that are in development to see whether they work as expected.

About this task

When sessions are submitted to the Event Tester, they are retained only during your current session. After you log out or are timed out, the session is no longer available in the Event Tester.

If you installed IBM Tealeaf CX RealiTea Viewer on your local desktop, you can choose to replay the session in RTV and then save it as a TLA file. This file can be uploaded at any time, again and again, to the Event Tester through the Event Tester's user interface.

Procedure

1. In the left column, click the **Send** to Event Tester icon.
2. In the dialog, enter a memorable name for the session, such as `Mobile Session`.
3. Click **Send to Event Tester**.
4. When prompted, you can choose to go to the Event Tester to explore how it works.
5. If you are done exploring, select **Configure** > **Event Manager** to return to the Event Manager to create more event objects.

Use groups and labels to store events and event-related objects

Before you begin building events and event-related objects, you should consider creating groups and labels to store them. You may build hit attributes, events, session attributes, and dimensions, so you should try to find ways to create consistent labels in each of the appropriate tabs in the Event Manager.

This table shows lists and describes one set of labels and groups for client framework data:

Label/Group	Description
`Source - UIC`	Event objects created to track data from Tealeaf IBM Tealeaf UI Capture
`Source - Android`	Event objects created to track data from the IBM Tealeaf Android SDK
`Source - iOS`	Event objects created to track data from the IBM Tealeaf iOS SDK
`Source - Mobile`	Event objects created to track data from mobile devices in general

Collect environmental data with step-based events

You create step-based events to collect client environment data. You create an event for each environment property in the message that you want to collect data on.

Process to create step-based events

To create step-based events you:

1. Create a step attribute
2. Create an event
3. Test the event
4. Repeat 1-3 for each additional environmental property that you want to collect data on
5. Create dimensions to store detected values from the events so you can generate reports for the events

Collect environment data from extended user agent parsing

If the Tealeaf Reference session agent is able to identify user agent information that is submitted from the client framework, values are populated in the `[ExtendedUserAgent]` section of the request. For example:

```
[ExtendedUserAgent]
TLT_BROWSER=Native
TLT_BROWSER_VERSION=Native8.5
```

```
TLT_BROWSER_PLATFORM=Android
TLT_TRAFFIC_TYPE=MOBILE_APP
TLT_BROWSER_JAVASCRIPT=
TLT_BROWSER_COOKIES=
TLT_BRAND=verizon
TLT_MODEL=verizon DROIDX
TLT_SCREEN_HEIGHT=854
TLT_SCREEN_WIDTH=480
```

Collect environment data from JSON Messaging

More device information is available in the client environment data that is submitted with each set of JSON messages.

When viewing a request in BBR that contains step attributes, click the **Click here to view Step Attributes** link at the top of the request. Step attribute data is displayed in easy-to-read format.

At the end of the step attributes is the `clientEnvironment` section. This example data is from a session that is initiated from an iOS device and captured by the iOS Logging Framework:

```
"clientEnvironment": {
        "mobileEnvironment": {
            "appName": "",
            "android": {
                "keyboardType": "NO_KEYS",
                "brand": "verizon",
                "fingerPrint": "verizon/shadow_vzw/
                                cdma_shadow:2.3.4/4.5.1_57_DX8-
                                51/120111:user/release-keys"
            },
            "totalMemory": 56512512,
            "totalStorage": 1711356,
            "orientationType": "PORTRAIT",
            "appVersion": "8.5.7.1",
            "manufacturer": "motorola",
            "userId": "mcbk83",
            "locale": "English (United States)",
            "deviceModel": "DROIDX",
            "language": "English"
        },
        "width": 480,
        "height": 854,
        "osVersion": "2.3.4"
    }
```

Top Movers

When you create events and dimensions that are associated with the client framework data set, you might want to create Top Movers to track deviations in values for these items. For example, if you created an event to track the value in the shopping cart at the time of checkout, you might want to create a Top Mover to track the variations in these values over time.

If you configured your Tealeaf solution to automatically create Top Movers for new events and dimensions, Top Movers might already be created for you.

After you gather environmental data

You created a number of event objects to locate and track important aspects of your mobile native application or applications that are monitored by one or more Tealeaf client frameworks.

You can now create reports from this data.

For specific uses, it is helpful to create and deploy search templates, which are configured to search for mobile native application sessions.

Creating a step attribute through BBR

Through Request view in BBR, you can review the JSON messages and then create attributes or events (and attributes) from them.

About this task

In this example, you create a step attribute that tracks total storage for the mobile devices.

When you create the event, you use the context menu, to create a step attribute to gather the value from the message property. Optionally, you can create an event to detect for the specific value in the request. At the end of each menu item, you can see the path to the message property.

Procedure

1. If you have the session open in BBR, click **Request**.
2. In the navigational panel, click the first page in the list.
3. Click the **Click here to view Step Attributes** link. The client framework messages are displayed in a more legible format.
4. Scroll down until you find the `clientEnvironment` section. You might notice that as you move a gray highlight bar appears on the name-value pair in the message under the mouse point.
5. Move the mouse over the name-value pair for the event you want to collect data for. For example, the `totalStorage` line. The gray bar highlights the name-value pair.
6. Right-click to open the **context menu**.
7. Select **Create New Step Attribute from selection**. The browser window that was displaying the session list is switched to display the Hit Attributes tab in the Tealeaf Event Manager. The message property information is loaded into the New Attribute dialog.
8. To simplify locating this item in a list, change the Name value to something you can remember. In this example, you might name the value: `Mobile - Environment - Total Storage`.
9. Click **Save Draft**.

Creating an Event without a replay session

You do not need a replay session to create an event, you can create an event from scratch.

About this task

In the event definition, you define the event, for example `Mobile - Environment - Total Storage`, to detect for the presence of the step attribute of the same name. When the event is triggered, the value that is recorded is the value in the step attribute. In this example, the attribute is evaluated after every step, tracked on the first match in a session, and has a numeric value.

Procedure

1. If you are not already in the Event Manager, in the **Portal** menu, select **Configure** > **Event Manager**.
2. Click the **Events** tab.
3. Click **New Event**.
4. Define the event properties:

   Property
   Description

   Name
   Set this value to the same as the step attribute. For example, `Mobile - Environment - Total Storage`.

   Description
   Add a user-friendly description, for example:

   ```
   from step attribute of same name
   ```

Evaluate
Select `After Every Step` from the drop-down.

Track
If this attribute does not change at any time, select `First per Session` from the drop-down.

Value Type
For the type of value to record, select `Numeric`.

5. In the Condition step, set the conditions for the attribute to be evaluated:
   a) In the left navigation panel, click the **Hit Attributes** category.

   Note: Step attributes are stored and referenced just like other hit attributes in Tealeaf.

   b) Select the attribute, for example, `Mobile - Environment - Total Storage`.
   c) From the first drop-down, select when the attribute is tracked, for example, `First Value`.
   d) From the second-drop down, select when the attribute is evaluated, for example, `Is not empty`.
6. Click the **Value** step. Configure the attribute value:
   a) Click **Select Item to Record...**.
   b) Click the **Hit Attributes** category.
   c) Select the attribute, for example, `Mobile - Environment - Total Storage`.
   d) From the drop-down, verify that the selected option is set to `First Match per Hit`.
7. Configure the other steps of your event as needed.
8. Click **Save Draft**.

Testing events to verify that they work as expected
After you identified a session with your event, you might find it useful to send to the Event Tester. In the Event Tester, you test events that are in development to see whether they work as expected.

About this task

When sessions are submitted to the Event Tester, they are retained only during your current session. After you log out or are timed out, the session is no longer available in the Event Tester.

If you installed IBM Tealeaf CX RealiTea Viewer on your local desktop, you can choose to replay the session in RTV and then save it as a TLA file. This file can be uploaded at any time, again and again, to the Event Tester through the Event Tester's user interface.

Procedure

1. In the left column, click the **Send** to Event Tester icon.
2. In the dialog, enter a memorable name for the session, such as `Mobile Session`.
3. Click **Send to Event Tester**.
4. When prompted, you can choose to go to the Event Tester to explore how it works.
5. If you are done exploring, select **Configure** > **Event Manager** to return to the Event Manager to create more event objects.

Saving your event to the server
After you test your attribute and event, save them to the server.

Procedure

1. From the **Portal** menu, select **Configure** > **Event Manager**.
2. To commit the change to the server, click **Save Changes** in the toolbar.
3. Enter a comment as needed. Click **Commit**.

Note: It might take a few moments for newly created step attributes or events to appear in the Event Manager. However, they are immediately applied to capture stream after they are committed to the server.

Creating dimensions from your mobile events

Whenever an event is triggered, the session attribute is updated, and any dimensions that are associated with the event are updated. A **dimension** is a piece of contextual information that is recorded when an event fires. You can use dimensions to create reporting filters to segment report along the type of client framework session.

About this task

For example, suppose that you created an event to track application error messages that are delivered to your visitors. While this kind of event is useful, it is important to be able to provide contextual information for the event. Does it occur in specific versions of the application? For which models? Is it localized to a specific locale? Answers to these questions can be useful to your developers in resolving the source of the issue. These bits of contextual information can be captured and stored in dimensions, the values for which are recorded when the event occurs.

One of the elements of the client environment data that can be tracked is the model of the device that is used to initiate the session. You can create an event to track the `deviceModel` attribute. In this example, you create a dimension to track the device model event, as captured by the `Mobile - iOS - Device Model` event.

Procedure

1. In the **Portal** menu, select **Configure** > **Event Manager**.
2. In the Event Manager, click the **Dimensions** tab.
3. In the Dimensions tab, click **New Dimension**.
4. The New Dimension dialog is displayed. Specify the properties:

   Property
   Description

   `Name`
   Provide a readily identifiable name.

   `Description`
   Provide a user-friendly description.
5. Click the **Select...** button and select the event that you created.
6. Specify the values to record:

   Property
   Description

   `Values to Record`
   For now, set this value to `Whitelist + Observed Values`.

   Turn On Logging
   Click this button to enable logging of values. Logged values are used to create whitelists, which are described below.
7. Leave the remaining values as their defaults.
8. Click **Save Draft**.
9. Click **Save Changes**.
10. Add a note to indicate the change. To commit your changes to the server, click **Save Changes**.

Associating the dimension with the event

About this task

To begin capturing values for the dimension, it must be associated with one or more events through a report group. A **report group** is an organizational structure that groups dimensions into a single reporting entity. It is the report group that is linked to the event; when the event fires, currently available values for any dimensions in the report group are recorded with the event.

Procedure

1. In the Event Manager, click the **Dimensions** tab.
2. In the Event Filter in the left panel, enter `Application Error` or another string that identifies the name of the event with which you want to associate the event.
3. When the event is displayed in the main panel, right-click it and select **Edit Event...**.
4. The Event wizard displays the event definition. Click the **Report Groups** step.
5. In the left panel, click **<New Report Group**.
6. Enter a meaningful name for the report group, such as `Mobile Device Information`.
7. Click **Add Dimensions...**.
8. Select the dimension that you just created.
9. Populate the other fields as needed.
10. Click **Save Draft**.
11. In the Event wizard, the report group and dimension are displayed in the Report Groups step.
12. Click **Save Draft**.
13. In the toolbar, click **Save Changes**.
14. Enter a comment and click **Commit**.
15. The dimension is now associated with the new report group, which is available in the Dimensions tab.
    - The report group was associated with your application error event, so whenever the application error event occurs, the value for the device model is recorded with the event.

Capturing dimension data

You configured your dimension to capture the device model information that is captured from an event. Currently, your dimension is configured to capture whitelisted values and observed values.

- **Whitelisted values** are values that you specifically configure the dimension to look for. When a value that you added to the dimension's whitelist occurs in the capture stream, the value is recorded as dimensional data for reporting purposes in the database.
- **Observed values** are any value detected in the capture stream. If the source of the dimension, an event in this case, occurs, then the observed value is detected and recorded into the database each time that it occurs.

  Note: When the capture of observed values is enabled, each and every detected instance of the dimension is recorded, which can quickly grow the size of the data that is stored in the database. If it is cleared, the data growth can consume all available disk space.

So, in the default configuration, which this event uses, all observed values are recorded as independent values. The number of device models, however, is finite, and potentially well-defined.

Note: As part of your use of this dimension, you should configure this dimension to use a whitelist after it is allowed to gather and log values for 24 hours. In this manner, you can begin to build your whitelist until you are ready to disable the capture of observed values, which greatly reduces the disk requirements for storing the dimension.

- For more information about how to manage the configuration of your dimensions to minimize storage requirements, see "Data Management for Dimensions" in the *IBM Tealeaf Event Manager Manual*.

Step-based eventing

Through *step-based eventing*, you can create Tealeaf events of these user interface events that are generated by your rich internet application.

On the traditional, HTML-based web, user actions typically triggered a single responding action from the web server. When you clicked a button, a form was submitted. When you clicked a link, a new page was loaded. For applications built on this framework, an individual event might occur only once per page.

In rich internet applications, however, this paradigm was altered. Many user interactions on a page do not change the page itself. In fact, a user can complete the same action of interest multiple times. For example, suppose that your web application enables the entry of multiple addresses from a single form. When **Submit** is clicked, the address data is submitted, and the form is cleared, enabling another entry. In this case, the same event, `SubmitAddress`, can occur multiple times on the same page. In Tealeaf, you want to be able to track all of these occurrences, instead of just the first one.

Note: A primary usage for step-based eventing is to track events that may occur multiple times on a single page.

With *step-based eventing*, you can create Tealeaf events of these user interface events that are generated by your rich internet application. In addition to creating events from individual hits, you can also create events from steps, which are individual user actions that are captured from the client application and submitted to Tealeaf by using one of Tealeaf's client frameworks.

- A step can be considered a "subhit" of a hit; a step reflects a discrete, trackable user action, or a server-side action that does not result from a user action (such as a redirect).
- Steps are captured by a client framework, which is bundled together, and submitted as JSON messages to Tealeaf. These messages are then inserted into a designated section of the request of the parent hit.

Step-based eventing enables the capture of multiple events from a single page of your client application.

Note: Step-based eventing requires licensing, installation, and configuration of one of the Tealeaf client frameworks, including IBM Tealeaf CX UI Capture for AJAX, IBM Tealeaf Android SDK, and IBM Tealeaf iOS SDK. Beginning in Release 8.5, new versions these frameworks are required to enable step-based eventing. For more information, contact Tealeaf Professional Services.

Note: IBM Tealeaf CX UI Capture for AJAX is only available to legacy users.

This information provides background information about client framework-generated steps and step-based eventing.

Technical definition of a step

A *step* is defined as a specially formatted JSON message that is submitted by the Tealeaf client frameworks to represent a session state of a form field.

- Step messages can contain any type of data. The data depends on the specific client framework that is sending the message.
- A step contains UI events from a single session only.
- In Tealeaf, these messages are submitted in JSON format and are not easy to decipher in raw format.

Step-based eventing

Step-based eventing has its own set of prerequisites, limitations, and message types. A *step* is defined as a specially formatted JSON message that is submitted by the Tealeaf client frameworks to represent a session state of a form field. Step messages can contain any type of data. The data depends on the specific client framework that is sending the message. A step contains UI events from a single session only. In Tealeaf, these messages are submitted in JSON format and are not easy to decipher in raw format.

Pre-requisites

To create step-based events, these components are required:

- Tealeaf Release 8.5 or later
- PCA Build 33xx. Tealeaf recommends using PCA Build 3330 at a minimum. Since PCA Build 3330, bug fixes and new features were added that can be of interest to you, including the ability to capture IPv6 addresses and support for new Linux platforms. You must be able to configure the capture of the `application/json` POST data types through the IBM Tealeaf CX Passive Capture Application.
- One or more of the Tealeaf capture solutions.

Limitations

The maximum length for selected values of text for attributes and events is 256 characters.

Distance and Sequence events operate on hits, not steps. As a result, the distance between events on multiple steps of the same hit evaluates to zero.

Message types

Events that are captured from client frameworks are bundled together and submitted as a set of messages. A *message* from a client framework is what defines a single step in Tealeaf, which is a single event that is identified and captured by a client framework.

Multiple messages can represent a single action of the visitor. For example, clicking a radio button might result in two messages of different types: one for the click event and one for change event.

Note: If you do not want to double count actions, use both the event type AND the ID/name when you create events for a specific action. If you look only for `ID = checkout method` for example, then this event fires twice when you only wanted it to fire once.

The volume of messages can depend on the configured logging level, which is defined in the client frameworks. UI Capture does not support dynamic logging levels.

Example raw request body

The `[RequestBody]` following information includes a sample raw request, which contains a set of JSON messages.

In the raw request, this example is a single paragraph. You cannot use the raw request body to create step-based attributes.
While it is possible to create hit attributes from the `[RequestBody]` section, it is not recommended, as this format might change over time.

```
[RequestBody]
{"version":"0.0.0.4","serialNumber":1,"sessions":[{"id":"ID14H2M3S663R0.36228193
267311725","startTime":1326837723663,"timezoneOffset":480,"messages":[{"type":2,
"offset":2226,"count":1,"context":{"type":"LOAD","name":"root","renderTime":
2226}},{"type":6,"offset":2230,"count":2,"exception":{"description":"Unable to
get value of the property 'nodeValue': object is null or undefined",
"url":"http://straussandplesser.com/store/js/coremetrics/eluminate.js",
"line":1}},{"type":4,"offset":24878,"count":3,"event":{"type":"click"},"target":
{"id":"[['main'],['DIV',1],['DIV',0],['TABLE',0],['TR',0],['TD',0],['DIV',0],
['P',0],['A',0]]","idType":-2,"type":"A"}},
{"type":2,"offset":24880,"count":4,"context":{"type":"UNLOAD","name":"root"}}]}]}
```

After the messages were passed through Tealeaf, the raw request is stored in the `[RequestBody]` section of the request, which is viewable through Request View in BBR.

Example formatted request body

When the JSON messages are received, Tealeaf reformats them into a more legible format. This information is available at the bottom of the request, which is formatted for view in Request View in BBR. This example shows a request with four separate messages in a `"messages": []` block:

```
{
    "version": "0.0.0.4",
    "serialNumber": 1,
    "sessions": [
        {
            "id": "ID14H2M3S663R0.36228193267311725",
            "startTime": 1326837723663,
            "timezoneOffset": 480,
            "messages": [
                {
                    "type": 2,
                    "offset": 2226,
                    "count": 1,
                    "context": {
                        "type": "LOAD",
                        "name": "root",
                        "renderTime": 2226
                    }
                },

                {
                    "type": 6,
                    "offset": 2230,
                    "count": 2,
                    "exception": {
                        "description": "Unable to get value of the property
                        'nodeValue': object is null or undefined",
                        "url":"http://straussandplesser.com/store/js/
                         coremetrics/eluminate.js",
                        "line": 1
                    }
                },

                {
                    "type": 4,
                    "offset": 24878,
                    "count": 3,
                    "event": {
                        "type": "click"
                    },
                    "target": {
                        "id":
                        "[['main'],['DIV',1],['DIV',0],['TABLE',0],
                         ['TR',0],['TD',0],['DIV',0],['P',0],['A',0]]",
                        "idType": -2,
                        "type": "A"
                    }
                },

                {
                    "type": 2,
                    "offset": 24880,
                    "count": 4,
                    "context": {
                        "type": "UNLOAD",
                        "name": "root"
                    }
                }
            ],
        }
    ]
}
```

Each message in the `"messages": []` block is demarcated by a set of curly brackets.

- Data that is defined at the same level as `messages` (such as `serialNumber` or `timezoneOffset`) is considered environmental data.
- Each step message is a single step.
- Step-triggered events can fire per message step.

- In the preceding example, there are four-step messages. As a result, step-triggered events can fire up to four times on this hit.

Each step-triggered event also has access to the hit attribute data of its parent hit and the environmental data included for reference in each step.

When you create step attributes, the value that is extracted is the contents between the colon (`:`) and the final comma (`,`) on the line.

Suppose that you want to monitor exception messages that are submitted from the client framework. Exception messages are type 6 messages. This example shows the type 6 message from the example:

```
{
    "type": 6,
    "offset": 2230,
    "count": 2,
    "exception": {
        "description": "Unable to get value of the property
        'nodeValue': object is null or undefined",
        "url": "http://straussandplesser.com/store/js/
                                coremetrics/eluminate.js",
        "line": 1
```

In this example, you can see that the exception message text is stored in the `description` value. To reference this value in step-based eventing, when you create the step attribute to monitor the message, the node in the tree is referenced by using the following structure:

```
sessions[0].message.exception.description
```

The naming structures for the `sessions` and `messages` nodes are changed, and the `type` identifier is omitted.

Note: When you create step attributes through BBR, you use the menu, which automatically pre-populates the attribute with the appropriate reference within the Event Manager.

Step-based objects

In Tealeaf, you can create two types of objects to monitor events that are captured from a client framework and passed as messages to Tealeaf:

- *Step attributes* are hit attributes that acquire its data from a step. Step attributes are specified in a slightly different manner but complete an identical function.
- *Step-based events* are standard Tealeaf events that are configured to fire on one of the steps triggers. As conditions, they can use any standard type of Tealeaf condition, and also step attributes.

Default step objects

Tealeaf provides a number of step-based events and attributes for use in step-based eventing.

- For more information about provided step attributes, see "Pattern Objects Reference" in the *IBM Tealeaf Event Manager Manual*.
- For more information about provided step-based eventing, see "EES Reference - Tealeaf Event Reference" in the *IBM Tealeaf Event Manager Manual*.

Step trigger types

To support step-based eventing, the Event Manager now provides two more trigger types:

Trigger
Description

`Every Step`
Event is evaluated with other events in each step.

`After Every Step`
Event is evaluated after every step is evaluated.

Note: This trigger is rarely used.

In the previous example, any event triggered to fire on `Every Step` is checked for each combination of JSON `message` and environmental data. In the previous example, any `Every Step` event is checked for the load, unload, exception, and other data message.

Note: Step attributes are permitted to reference objects from the parent hit. As a result, you can reference hit attributes in step events, but not vice versa.

In the event definition, the trigger can be selected from the Evaluate drop-down:

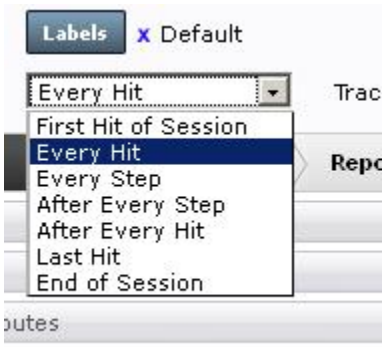

Figure 1. Available event triggers

Available triggers are displayed in the order of evaluation. For a particular hit with underlying steps, each `Every Hit` event is evaluated first, followed by each `Every Step` event and `After Every Step` event. Then, the `After Every Hit` events are evaluated.

Note: The events that fire on each trigger determine the availability of data. An event can use data from any event that fired before the current event. In a multi-hit session, the `After Every Hit` trigger fire on the previous hit before the events configured to fire on `Every Hit` from the next hit. The same applies to step-based triggers.

The order of firing is more accurately displayed as a nested structure:

```
* First Hit of Session
    * Every Hit
        * Every Step
        * After Every Step
    * After Every Hit
    * Last Hit
* End of Session
```

Considerations for using the `After Every Step` trigger
In almost all cases, when you create step attributes, you are interested in the current context of the session. You create attributes to monitor the current data that is available as of the current step. As a result, the `After Every Step` trigger is rarely used.

In the example below, the `After Every Step` trigger is used. This scenario mirrors the `After Every Hit` trigger usage, except that it applies to steps instead of hits.

The `After Every Step` trigger is useful when you must compare the current state with the previous state. For example, suppose you want to know whether users clicked the same object twice in a row.

- To test this scenario, you must know both the object currently being click, and the previously clicked object. If the events that track both the current and previous states fire on the same trigger, they are updated at the same time and therefore always have the same value.

- However, if the previous state event fires just after the current state value by using the `After Every Step` trigger, the previous state event is not updated when the current state event fires. Therefore, you can compare the current state with the previous state by using an event that fires on the `Every Step` trigger.

Note: Form messages contain the `currState` and `prevState` properties within a step. The `currState` property refers to the final value of the form field after editing, and `prevState` refers to the default value before editing. These references do not work for testing if the same action occurred twice, since the default value can be reset to blank each time it is accessed.

Privacy

To manage blocking or masking of sensitive data, Tealeaf provides privacy mechanisms to manage specific data before it is transmitted to Tealeaf.

Note: Application of privacy blocking or masking in the PCA or in the Windows pipeline requires complex regular expressions, which can cause significant performance degradation if improperly specified. Tealeaf strongly recommends using the privacy solution that is provided with your client framework to manage sensitive data.

Browser based replay and step-based events

In Browser based replay, steps are displayed as subpages to the main page on which they occurred.

Any triggered events are displayed beneath them.

During replay, steps can be displayed in a more user-friendly format.

Note: Replay of step-based events in RTV is not supported.

A single web action can require multiple attributes and events to track. You can create multiple attributes, which are inputs to a single compound event to track a single user action.

Navigable Pages List

Using the Request view of BBR, you can create hit attributes and events for steps.

When you load a session that contains JSON-based steps into BBR, the Navigable Pages List looks like:

In the preceding image, the step that is captured from the visitor's user interface are indicated by the `UIEvent` label. In the preceding example, each instance also lists the range of user interface events captured. `UIEvent: 1 -4` indicates that the specific step includes 4 individual user interface events.

Figure 2. BBR Navigable Pages List

Navigable Pages List

The Navigable Pages list shows the events that were captured in a user's session. You can use the events to create hit attributres and events for steps.

Through the request view of BBR, you can create hit attributes and events for steps. When you load a session that contains JSON-based steps into BBR, the Navigable Pages List looks like:

A step that is captured from the visitor's user interface is indicated by the `UIEVENT` label. Each instance also lists the range of user interface events captured. `UIEVENT: 1 -4` indicates that the specific step includes 4 individual user interface events.

Viewing formatted JSON messages

When one of the UIEvent steps is selected, you can review the JSON messages that are submitted as part of the step.

About this task

Note: Any BBR user can view the formatted client framework messages. To create attributes and events from them, you must have access permissions to the Tealeaf Event Manager.

Procedure

1. Click **Request** in the toolbar to display the request data.
2. Click one of the `UIEvent` entries in the Navigable Pages List.
3. The raw JSON messages are displayed in the `[RequestBody]` section.
4. However, this information is not easy to read. To review the JSON messages in a more legible format, click the **Click here to view Step Attributes** link at the top of the request pane.
5. The list of JSON messages are broken out into separate lines for easier reading.
   - Each selectable item is a name-value pair that is highlighted when you move your mouse over it.
   - Null values can be selected for creation of step-based attributes.
   - Hit and event objects that you create search for the values for the specified JSON item.

Event manager processing of step-based event objects

You can create step attributes through Request view in Browser Based Replay.

Note: Creation of step attributes is not supported in RTV.

When you create objects through Browser Based Replay, the Event Manager checks to see if the selected content is already referenced in an existing event object. If so, the Event Manager selects that object for you to edit.

In some cases, the selected event object is provided by Tealeaf and is therefore not editable. For example, Tealeaf provides the `CUI Hit` hit attribute, which references the contents of the HTTP_X_TEALEAF request variable. When you choose to create attributes or events from the values of this request variable, the Event Manager selects the `CUI Hit` attribute for you to edit. This hit attribute cannot be edited.

Note: If you want to create more step attributes and events from session data for which attributes or events are already created, you must create them manually through the Event Manager.

Note: There is a known issue in which the PCA fails to properly recognize UTF-8 encoding in data that is submitted from client frameworks, and the data can be mangled in the stored session, causing issues in eventing and search.

Note: The following information applies to IBM Tealeaf version 9.0A only.

9.0A can properly recognize UTF-8 encoding in data that is submitted from client frameworks.

Permissions for creating step-based event objects

To create step attributes, you must have permissions to access the Tealeaf Event Manager, where event-related objects are created in the Portal.

To test access, select **Configure** > **Event Manager**

Required access

Note: To create step attributes, you must have permissions to access the Tealeaf Event Manager, where event-related objects are created in the Portal.

To test access, select **Configure** > **Event Manager**.

BBR step attribute context menu

Use the BBR step attribute context menu to create a new event from a step attribute or to create a new step attribute.

About this task

Note: To access the BBR step attribute context menu, you must have access to the Event Manager.

Procedure

1. Select a page from the page navigation list in BBR.
2. Right-click a name-value pair in the formatted JSON message.
3. From the context menu, select one of the following options:
   - **Create New Event from Step Attribute selection** to create a step-based event and any necessary hit attribute to gather the data.
   - **Create New Step Attribute from selection** to create a step attribute.

Creating a step attribute

When you select a JSON item in BBR and choose to create a step attribute, the Event Manager is opened in the browser window currently opened to the Portal.

The following dialog is displayed:

Note: Depending on your browser type and configuration, you can manually switch over to the **Portal** window.

Figure 3. Creating a step attribute

Step attributes are commingled with hit attributes. They do not belong to a special category. What defines an attribute as a step attribute are the properties that are listed.

Table 8. Key Properties:

Attribute Properties	Description
`Use Step Pattern`	For step attributes, the Use Step Pattern radio button is selected for you, which enables specification of the xpath to the node whose value you want to track. • When a step pattern is used to identify an attribute, the check is completed by using a case-sensitive search by default. If you choose to change the type of pattern tag to a step pattern, the existing case-sensitive settings are preserved.
`Step Attribute Path`	The Step Attribute Path value contains the node information to uniquely identify the JSON value to acquire in the attribute. For the preceding example, the path is: `.sessions[0].message.exception.description` This provides a unique path to the description value for the exception message that was submitted from a client framework to Tealeaf.

Note: You can complete the same Post-Match Operations on a step attribute that you can complete on a hit attribute.

Data format
The values of step attributes are always treated as text patterns. As a result, operators such as `equals` perform text-based comparisons, even if the captured value is a numeric or Boolean value.

Data availability
Like the hits that contain them, steps are processed in isolation from all other steps. For example, if you want to use data from step 1 for use on step 2, you must create an event to record the data from step 1 for later use.

- Since each step is associated with a parent hit, any hit attributes triggered on the parent hit are available for reference in each step of the hit.
- However, step attributes are available only within the single step that is being evaluated.
- If you want to use a hit attribute in a step attribute, the event trigger must be configured to be evaluated on one of the step triggers.
- Data from events that are triggered on previous steps is available in later steps.

Using data between step attributes
A step triggered event uses only data that is contained in the step in which it is triggered, which is a similar behavior to how hits are triggered.

To use data from step 1 in step 2, you must record the data from step 1 in an event and then reference the event in step 2.

For example, suppose your request data for a single hit looks like:

```
[appdata]
TLT_URL=/tealeaftarget.php
TLT_CUI_URL= /checkout

[StepAttributes]
                {
                    "type": 4,
                    "offset": 8063,
```

```
                    "count": 1,
                    "event": {
                        "type": "change"
                    },
                    "target": {
                        "id":  firstname",
                        "idType": -1,
                        "type": "INPUT",
                        "dwell": 2196,
                        "currState": {
                            "value": "MyName"
                        }
                    }
                },
                {
                    "type": 4,
                    "offset": 2293,
                    "count": 2,
                    "event": {
                        "type": "click"
                    },
                    "target": {
                        "id": "login:guest",
                        "type": "INPUT",
                        "subType": "radio",
                        "currState": {
                            "checked": true,
                            "value": "guest"
                        }
                    }
                },
```

In the preceding data:

- `[appdata]` data is available through standard hit attributes.
- There are 2-step messages:
  - Step 1: The first step identifies the `change` client event, in which the `firstname` form field is set to `MyName`.
  - Step 2: The second step identifies the `click` client event, in which the `login.guest` element is set to `guest`.

A single step-triggered event cannot use data from both step 1 and step 2 at the same time. For example, you cannot create a step-triggered event that fires on the click message and records the value of the `firstname` value by using only step attributes.

To capture the value of Step 1 based on the condition of Step 2, you must:

- Create a step attribute to capture `firstname`'s value on Step 1.
- Create an event that records the value of the step attribute for Step 1.
- Create an event that fires on the click for guest and uses the value for the Step 1 event for the guest value.

Important notes on step-based eventing

- A single web action can require multiple attributes and events to track. You can create multiple attributes, which are inputs to a single compound event to track a single user action.

Capturing a specific value
By default, a step attribute captures all possible values for the selected JSON path. When the attribute is specified, any value that is detected for the node becomes the value for the attribute.

About this task

In some situations, you can gather in the step attribute only specified values. For example, suppose that you are tracking the following JSON path:

```
.sessions[0].message.clientState.event
```

By default, any step attribute can capture any instance of any value. So, your attribute can capture values such as `load`, `attention`, `resize`, or `scroll`. Suppose that you are interested in creating a step attribute to track only the `scroll` values. After you create the step attribute through BBR, you can complete the following modifications to the attribute definition through the Event Manager.

Procedure

1. Edit the step attribute.
2. Click the**Post Match Operations caret**.
3. Select the **User RegEx** check box.
4. In the RegEx textbox, enter:

   ```
   scroll
   ```

5. To save the change, click **Save Draft**.
6. To commit the change, click **Save Changes**.

Results

Now, the step attribute records only the instances of the `scroll` value for the specified JSON path.

As an alternative, you can specify a step attribute without using the RegEx portion. When you use the step attribute in an event, specify that the value of the step attribute equals `scroll`.

Creating a step event
When you select a JSON item in BBR and choose to create an event, the Event Manager is opened in the browser window currently opened to the Portal. The Event Wizard is displayed.

Note: Depending on your browser type and configuration, you can manually switch over to the **Portal** window.

The default event checks every step to see whether the JSON item is present and records the last occurrence in the event by default. Using this configuration, you can track the number of sessions in which the event occurred.

Note: If you cancel creation of a step-based event, you must revert the step attribute, if created, through the Hit Attributes tab.

Triggers for step objects
Step-based events can be evaluated on the `Every Step` and `After Every Step` trigger.

Triggers for compound events using step-based events as conditions
There are specific triggers that must be applied when compound events use step-based events as conditions.

Note: If you are creating an event with multiple conditions that uses one or more step-based events, you must set the event to be evaluated on `After Every Hit`. That trigger is evaluated after `Every Hit`, `Every Step`, and `After Every Step`, in that order.

Tracked Occurrences for step events
For any event that is triggered off step-based data, you must configure it to track occurrences at the individual hit level.

Note: Do not use session-level tracking, as those options operate only on the first or last hit of the session.

Condition step
When you create an event to track a JSON message item, the step attribute that is required to detect the name-value pair is also created in draft mode for you. For such scenarios, the Hit Attribute condition specifies the step attribute that the Event Manager has also created for you.

The Event Manager pre-populates the event definition with properties to identify the specific JSON item to track.

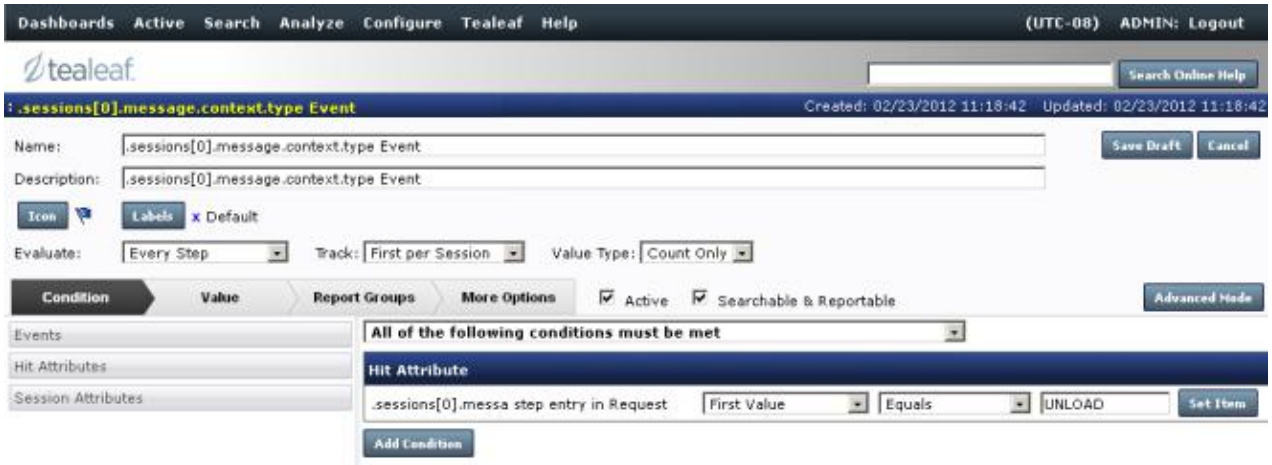

Figure 4. Creating a step-based event - Condition step

Note: For step attribute conditions, `Match Count` and `Last Value` value tests are not useful, as there is only one unique match and its value for a specified property on the hit.

Note: If you do not want to double count actions, use both the event type and the ID/name when you create events for a specific action.

- If you look only for `ID = checkout method`, then this event fires twice when you only wanted it to fire once. Suppose you want to track clicked objects. Each object has the event type `click`.
- To track clicks of a specific object, you must specify both the event type and object ID.
- If a step attribute with the same properties exists, the Event Manager uses the existing step attribute.
- As needed, you can add extra conditions to the event you are creating.

Table 9. Key Properties:

Attribute Properties	Description
Icon	You must select an identifying icon for your step-based events.

Table 9. Key Properties: (continued)	
Attribute Properties	**Description**
Labels	You can organize your step-based events into labels within the Event Manager.
`Evaluate`	Set the trigger to be either of the step-based triggers.
`Track`	Set the occurrences to track to monitor the first or last occurrence in the session or every occurrence. • If you want to track the number of sessions in which the event occurrence, set the value to `Last Occurrence`. • If you want to track each time that the event occurred in a session, set the value to `Every Occurrence`.
`Value Type`	Step-based events can track numeric or text values or the count of occurrences of the event.

Value step
The value of step-based events can be specified like any other event.

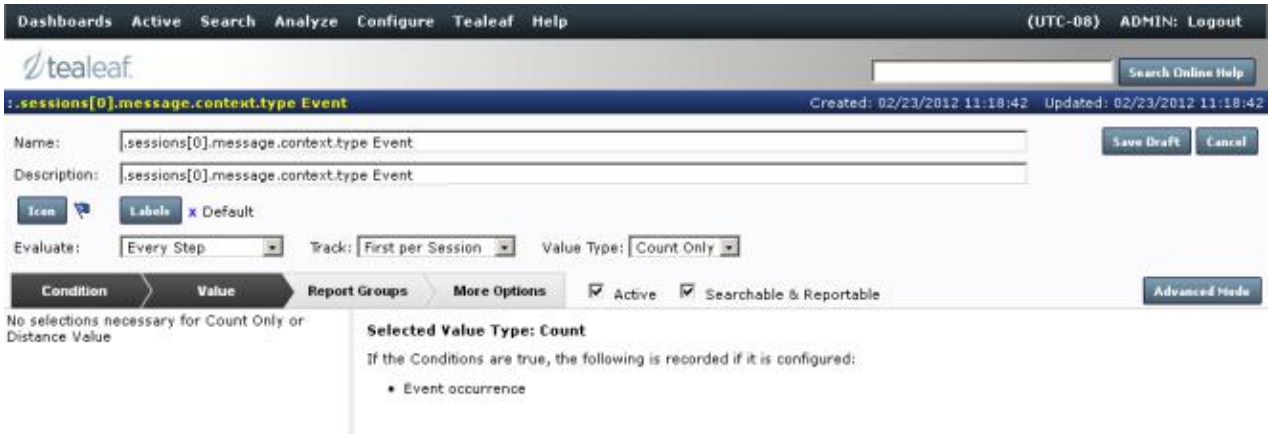

Figure 5. Creating a step-based event - Value step

Other steps
For step-based events, you can configure the other steps as you would any other event.

Advanced Mode
To see the native JavaScript created for your step-based event, click **Advanced Mode**.

Creating a dimension
After you create the step attribute, event, or both to track a value in a submitted message, you can create a dimension to record values from the event or attribute in the standard manner.

Note: Dimensions that are populated by step attributes or events can capture new values from multiple steps in a hit.

Support statement for creating step attributes and events in RTV
Creation of step attributes and events is not supported in RTV.

In request view, you can review the raw JSON messages in the `[RequestBody]` section of the request.

You can use RTV to save test TLA sessions, which can be loaded into the Event Tester to use as test data for step-based events.

In Event Tester
In the Event Tester, you can validate the triggering of step-based attributes and events. Step attributes and the events that are triggered from them are displayed as regular hit attributes and events in the test results.

Note: After you create step-based objects, it can take a few minutes before they are available for selection in the Event Tester.

Indexing and step-based events

Step-based event data is not indexed by default. You can, however, search for events through BBR and RTV.

It is possible to move data from one location in the request into another section which is automatically indexed for search.

Note: If you must index some JSON-based session data for search, you must use a privacy rule to insert the data into the `[appdata]` section. Creation of the rule requires configuration of a regular expression to locate the data. Regular expressions are considered an advanced configuration option, as if they are poorly specified, they can significantly impact system performance.

- For more information about configuring regular expressions, contact Tealeaf http://support.tealeaf.com.
- Use of privacy rules against JSON message data is likely to be supported in a later release.

Reference information about BBR and Events

Use the information here to find the IBM Tealeaf publications that contain information about BBR and Events.

Section
Description

"CX Browser Based Replay" in the *IBM Tealeaf cxImpact User Manual*
BBR documentation

"Browser Based Replay Interface" in the *IBM Tealeaf cxImpact User Manual*
How to use BBR, including how to access request view

"TEM Hit Attributes Tab" in the *IBM Tealeaf Event Manager Manual*
How to create hit attributes or step attributes in Event Manager

"TEM Events Tab" in the *IBM Tealeaf Event Manager Manual*
How to create events in Event Manager

"UI Capture for Ajax Guide" in the *IBM Tealeaf UI Capture for Ajax Guide*
Reference guide for the IBM Tealeaf CX UI Capture for AJAX solution

"Tealeaf Android Logging Framework Reference Guide" in the *IBM Tealeaf Android Logging Framework Reference Guide*
Reference guide for the IBM Tealeaf Android SDK

"Tealeaf iOS Logging Framework Reference Guide" in the *IBM Tealeaf iOS Logging Framework Reference Guide*
Reference guide for the Tealeaf IOS Logging Framework

"Event Tester" in the *IBM Tealeaf Event Manager Manual*
Portal-based Event Tester displays step-based attributes and events transparently

"Searching Session Data" in the *IBM Tealeaf cxImpact User Manual*
Searching for sessions through the Portal

"RealiTea Viewer - Request View" in the *IBM Tealeaf RealiTea Viewer User Manual*
Request view page for RTV.

Note: You cannot create step attributes through RTV.

"RealiTea Viewer - Session Search and Subsearch" in the *IBM Tealeaf RealiTea Viewer User Manual*
Searching for sessions through RTV

"Configuring CX Indexing" in the *IBM Tealeaf CX Configuration Manual*
How sessions are indexed and how data is added for indexing

"Privacy Session Agent" in the *IBM Tealeaf CX Configuration Manual*
Session agent that is used to move content in the request

Eventing for cxOverstat

IBM Tealeaf cxOverstat enables the capture of usability information from the visitor's web experience, as detected in the client and transmitted to Tealeaf. This information is anonymously collected for reporting purposes. Events can be used to specify specific conditions, such as a period of time, to record information.

Note: IBM Tealeaf cxOverstat is a separately licensable product of the IBM Tealeaf CX platform. For more information, contact your representative.

When the IBM Tealeaf cxOverstat JavaScripts are deployed on pages or ScreenViews of your web application, usability data such as X-Y location, relative location, and focal point are captured and transmitted to Tealeaf. A provided set of events is designed to capture the usability data.

- You can optionally associate other events with usability dimensions by using the report group templates that are provided by Tealeaf.

In this section, you learn more about the usability data that is captured for IBM Tealeaf cxOverstat, the data objects that are provided by Tealeaf to record them, and other eventing possibilities for usability data.

- The maximum length for selected values of text for attributes and events is 256 characters.

cxOverstat usability data

IBM Tealeaf cxOverstat data is submitted from the IBM Tealeaf UI Capture solution, when it was enabled for IBM Tealeaf cxOverstat.

IBM Tealeaf cxOverstat data comes in the following forms:

Type
Description

Comparison Analytics
With the Comparison Analytics overlay, you can view selected metrics for a web page and apply specific segments and filters to create customized reports.

- For more information about this feature, see "Comparison Analytics Overlay" in the *IBM Tealeaf cxOverstat User Manual*.

Heat mapping
Identifies regions on a page or ScreenView where visitors click, regardless of whether an object was selected.

- For more information about this feature, see "Using Heat Maps" in the *IBM Tealeaf cxOverstat User Manual*.

Attention mapping
Identifies the regions on a page or ScreenView that are most frequently displayed within the visitor's browser window.

- For more information about this feature, see "Using Attention Maps" in the *IBM Tealeaf cxOverstat User Manual*.

Form
Identifies visitor activities on forms on a page or ScreenView.

- For more information about this feature, see "Using Form Analytics" in the *IBM Tealeaf cxOverstat User Manual*.

Link

Identifies links that are most frequently selected by visitors to a page or ScreenView.

- For more information about this feature, see "Using Link Analytics" in the *IBM Tealeaf cxOverstat User Manual*.

Usability Eventing

To support the tracking and recording of usability activities on your web application, IBM Tealeaf provides a set of event objects to process usability data that is submitted in JSON format from the client framework.

For more information about the schema, see "Tealeaf JSON Object Schema Reference" in the *IBM Tealeaf Client Framework Data Integration Guide*.

- For more information about the properties that pertain to IBM Tealeaf cxOverstat, see "Tealeaf JSON Properties" in the *IBM Tealeaf Client Framework Data Integration Guide*.

Usability eventing

To support the tracking and recording of usability activities on your web application, IBM Tealeaf provides a set of event objects to process usability data that is submitted in JSON format from the client framework.

cxOverstat objects must be enabled

In addition to providing the standard reporting and search capabilities, IBM Tealeaf cxOverstat events are also used by the usability application itself. For example, when the Tealeaf user chooses to display the Form Analytics overlay in Browser Based Replay, the usability components in BBR query the Reporting database for the event data that pertains to form analysis for the current page or ScreenView.

Note: For correct capture and display of usability data in the IBM Tealeaf cxOverstat overlays in Browser Based Replay, all hit attributes, events, and dimensions must be enabled through the Event Manager. By default, these objects are installed and enabled. Disabling can disable usability functions and data.

Goal Based Dimensions

By default, Tealeaf publishes event data as soon as the Canister detects and processes it. IBM Tealeaf cxOverstat events, however, are configured to publish their event data at the end of the session. The event values that were detected at the time the event fired are recorded, but the recording does not occur until the session is completed by the user, or session timeout. This method for event publishing enables the support of the capturing of the dimension values associated with the event from their last occurrence in the session.

For example, suppose you use a dimension to capture whether a purchase was made. Since that information is not known until the end of the session, you can delay the publishing of IBM Tealeaf cxOverstat events and their related dimensions until the end of the session so the definitive answer (Yes or No) is captured in the dimension value.

Note: All IBM Tealeaf cxOverstat events are configured to have their data published at the end of the session. This setting cannot be modified for these events.

See "Goal Based Dimensions" in the *IBM Tealeaf Event Manager Manual*.

Data storage

After the IBM Tealeaf cxOverstat step attributes and events are triggered, they are recorded in the standard locations for Tealeaf.

Session

When the Event Engine detects an event, it records the event as data in the request of the hit on which the event occurred, along with any related dimensional data. IBM Tealeaf cxOverstat events are recorded in this manner.

Below is an example recording of the event + dimension combination in a request:

- These elements of data are written to the bottom of the request for each event that is triggered on the hit.

```
[TLFID_359]
Searchable=True
TLFID=359
TLFactValue=200
TLDimHash1=C6F8B06175B0630687EB80DF913A30CE
TLDimHash2=B0437419D4F0575FABDB726AAE61039C
TLDimHash3=3CFBA54F6873DFD55B0B09D32910B20E
TLDimHash4=0BDB5F014A7574C3B6DCCAD319321FED
TLDimHash5=7954797EAEBD4BD8B816EA63AF1CE05A
TLDimHash6=7954797EAEBD4BD8B816EA63AF1CE05A
TLDimHash7=7954797EAEBD4BD8B816EA63AF1CE05A
TLDimHash8=7954797EAEBD4BD8B816EA63AF1CE05A
TLDim1=/store/defaultpage
TLDim2=www.straussandplesser.com
TLDim3=store
TLDim4=206.169.17.19
TLDim5=TLT$NULL
TLDim6=TLT$NULL
TLDim7=TLT$NULL
TLDim8=TLT$NULL
```

For the listed event + dimension combination (`[TLFID_359]`), the event value (`TLFactValue`) is recorded for the Searchable event.

- Beneath the event value, each possible dimension value is listed in both hashed (`TLDimHash`) and standard (`TLDim`) form.
- Hashed dimension values are stored to support searching for dimension values that are longer than 32 characters. See "Searching Session Data" in the *IBM Tealeaf cxImpact User Manual*.
- For more information about request view in BBR, see "CX Browser Based Replay" in the *IBM Tealeaf cxImpact User Manual*.
- For more information about the data that is stored for events and dimensions in the request, see the **[TLFID_**] section in "RealiTea Viewer - Request View" in the *IBM Tealeaf RealiTea Viewer User Manual*.

Database

Usability data is stored as event counts of the predefined usability events. This data is stored in the Reporting database as normal event data. When a page or ScreenView for stored usability data is displayed in BBR, the application queries the Reporting database for the correct event counts to display in the selected overlay

Session

When the Event Engine detects an event, it records the event as data in the request of the hit on which the event occurred, along with any related dimensional data. IBM Tealeaf cxOverstat events are recorded in this manner. Below is an example recording of the event + dimension combination in a request:

- These elements of data are written to the bottom of the request for each event that is triggered on the hit.

```
[TLFID_359]
Searchable=True
TLFID=359
TLFactValue=200
TLDimHash1=C6F8B06175B0630687EB80DF913A30CE
TLDimHash2=B0437419D4F0575FABDB726AAE61039C
TLDimHash3=3CFBA54F6873DFD55B0B09D32910B20E
TLDimHash4=0BDB5F014A7574C3B6DCCAD319321FED
TLDimHash5=7954797EAEBD4BD8B816EA63AF1CE05A
TLDimHash6=7954797EAEBD4BD8B816EA63AF1CE05A
TLDimHash7=7954797EAEBD4BD8B816EA63AF1CE05A
TLDimHash8=7954797EAEBD4BD8B816EA63AF1CE05A
TLDim1=/store/defaultpage
TLDim2=www.straussandplesser.com
TLDim3=store
TLDim4=206.169.17.19
TLDim5=TLT$NULL
```

```
TLDim6=TLT$NULL
TLDim7=TLT$NULL
TLDim8=TLT$NULL
```

For the listed event + dimension combination (`[TLFID_359]`), the event value (`TLFactValue`) is recorded for the Searchable event.

- Beneath the event value, each possible dimension value is listed in both hashed (`TLDimHash`) and standard (`TLDim`) form.
- Hashed dimension values are stored to support searching for dimension values that are longer than 32 characters. See "Searching Session Data" in the *IBM Tealeaf cxImpact User Manual*.
- For more information about request view in BBR, see "CX Browser Based Replay" in the *IBM Tealeaf cxImpact User Manual*.
- For more information about the data that is stored for events and dimensions in the request, see the **[TLFID_**] section in "RealiTea Viewer - Request View" in the *IBM Tealeaf RealiTea Viewer User Manual*.

Database

Usability data is stored as event counts of the predefined usability events. This data is stored in the Reporting database as normal event data.

When a page or ScreenView for stored usability data is displayed in BBR, the application queries the Reporting database for the correct event counts to display in the selected overlay.

cxOverstat step attributes

To facilitate the tracking of usability data that is submitted from UI Capture, Tealeaf provides a set of step attributes to track this information.

- When IBM Tealeaf cxOverstat is licensed and enabled, these attributes are automatically enabled, and Tealeaf is tracking usability data.

  Note: If you are licensing IBM Tealeaf cxOverstat for an existing Tealeaf installation, you must use the Tealeaf Upgrader to install the IBM Tealeaf cxOverstat event objects. See "cxOverstat Installation and Configuration" in the *IBM Tealeaf cxOverstat User Manual*.
- A step attribute is a hit attribute that is configured to track values that are stored in JSON data that is submitted from a client framework. See "Step-Based Eventing" in the *IBM Tealeaf Event Manager Manual*.
- In the Events tab, these step attributes are contained in the `System Step Attributes`.
- For more information about the step attributes pertaining to IBM Tealeaf cxOverstat, see "Tealeaf JSON Properties" in the *IBM Tealeaf Client Framework Data Integration Guide*.
  - For more information about all event objects that are provided by Tealeaf, see "Tealeaf Standard Event Object Reference" in the *IBM Tealeaf Event Manager Manual*.

cxOverstat events

Usability events require dimensional data in order for them to properly function. The following events are provided by Tealeaf for IBM Tealeaf cxOverstat, and associated with each of these events is a set of pre-configured dimensions.

- In the Events tab, these events are contained in the `System Step Events`.
- Building block events are not displayed in the Portal at all.
- By default, the non-building block events for IBM Tealeaf cxOverstat are configured to be displayed in the Portal for search and reporting purposes. However, they are configured to not be displayed in session lists in the Portal, which includes QuickView and the Page List. While you can modify the event definitions to display them in these views, they are likely to be displayed in every session and thus clutter the display. See "TEM Events Tab" in the *IBM Tealeaf Event Manager Manual*.

Note: Except to add or remove dimensions that you created, do not edit these System Step Events. Do not edit the event definitions or remove any of dimensions that are listed for each event.

Primary Reporting Events
Usability attention map view time, usability click, and usability form field visit are the primary events that are used for the reporting of IBM Tealeaf cxOverstat data.

Table 10. Primary Reporting Events

Event Name	Description	Default Dimensions
Step - Usability Attention Map Y View Time	Attention View Time (Y) Event for Usability data	• ScreenView URL • Step - ScreenView • Step - Usability Focal Slice Y • Step - Usability View Port Height
Step - Usability Click	Click event for Usability data. This event is used for comparison analytics, heat maps, and link analytics data.	• Step - ScreenView URL • Step - ScreenView • Step - Target ID • Step - Target Relative XY
Step - Usability Form Field Visit	Field Visits + Dwell time for Event for Usability data	• Step - ScreenView URL • Step - ScreenView • Step - Target ID

By using viaTealeaf, you can access a set of reports that are configured for the usability data that is submitted to Tealeaf.

Building block events
A "building block" event is available in the Short Term Canister while the session is being captured. The event expires when the session is closed and moved to the Long Term Canister. Building block events cannot be used in search or reporting.

See "TEM Events Tab" in the *IBM Tealeaf Event Manager Manual*.

The following table describes the building block events that are defined. These events are unlikely to need modification.

Table 11. Building Block Events

Event Name	Description	Default Dimensions
Step - ScreenView (BB)	(Building Block) Latest ScreenView from ScreenView LOAD message	None
Step - ScreenView URL (BB)	(Building Block) Latest URL from ScreenView LOAD message	None
Step - Usability Attention Map Viewport Height	Normalized Viewport height (min of Viewport or Page height) **Note:** This event can be reviewed and modified in JavaScript only.	None
Step - Usability Focal Slice Y (BB)	(Building Block) Focal Slice BB event for Usability data **Note:** This event can be reviewed and modified in JavaScript only.	None

Table 11. Building Block Events (continued)		
Event Name	**Description**	**Default Dimensions**
Step - Usability Target ID + Type (BB)	(Building Block) Combines Target ID and ID Type into a single string **Note:** This event can be reviewed and modified in JavaScript only.	None

- For more information about the events that pertain to IBM Tealeaf cxOverstat, see "Tealeaf JSON Properties" in the *IBM Tealeaf Client Framework Data Integration Guide*.
  - For more information about all event objects that are provided by Tealeaf, see "Tealeaf Standard Event Object Reference" in the *IBM Tealeaf Event Manager Manual*.

cxOverstat dimensions

To capture IBM Tealeaf cxOverstat data for reporting purposes, Tealeaf provides a set of dimensions, which are used to store client usability data.

To locate these dimensions, enter `Step` in the filter text box in the **Dimensions** tab.

Table 12. cxOverstat Dimensions		
Dimension Name	**Description**	**Source Event**
Step - ScreenView	Records the latest Screenview for each Screenview LOAD message	Step - ScreenView [BB]
Step - ScreenView URL	Records the latest URL for each Screenview LOAD message	Step - ScreenView URL [BB]
Step - Target ID	ID of object that is acted on	Step - Usability Target ID + type [BB]
Step - Target Relative XY	Records the relative position of the action that is based on the object that is acted on	Step - Target Relative XY
Step - Usability Focal Slice Y	Records the Y focal slice. • Dimension includes a predefined whitelist of values for bucketing of slices for reporting.	Step - Usability Focal Slice Y [BB]
Step - Usability View Port Height	Viewport height of the browser window that explores usability overlays • Dimension includes a predefined whitelist of values for bucketing of slices for reporting.	Step - Usability Attention Map Y View Time

For more information about the dimensions that pertain to IBM Tealeaf cxOverstat, see "Tealeaf JSON Properties" in the *IBM Tealeaf Client Framework Data Integration Guide*.

For more information about all event objects that are provided by Tealeaf, see "Tealeaf Standard Event Object Reference" in the *IBM Tealeaf Event Manager Manual*.

cxOverstat report groups

IBM Tealeaf cxOverstat provides a set of report groups that contain the provided IBM Tealeaf cxOverstat dimensions. These report groups are predefined to contain only the dimensions that are required for IBM Tealeaf cxOverstat.

The following list displays the dimensions:

- Usability - Attention Map
- Usability - Click
- Usability - Form Analytics

IBM Tealeaf recommends creating separate report groups for use with the IBM Tealeaf cxOverstat system dimensions and then add dimensions as needed. These custom report groups can be added, modified, and removed as needed.

Note: After a dimension is added to any report group and committed to the database, it cannot be removed. Since you are not allowed to remove the default IBM Tealeaf cxOverstat report groups from the IBM Tealeaf cxOverstat events, any dimensions added to these report groups are retained permanently.

cxOverstat report group templates

For IBM Tealeaf cxImpact, the base report group template is `Standard`. This report group template supports up to four dimensions of your choosing.

For IBM Tealeaf cxOverstat, if you want to associate the usability data with other dimensions for cross-dimensional reporting, you can use report group templates as the basis for creating new report groups. A report group template is a template that contains pre-configured dimensions for a new report group when it is created. The report group template can be selected when you create a report group.

Note: You can create multiple of custom report groups from these templates. Only report groups that are based on these associated templates are available for usability events.

Note: You cannot make report group templates.

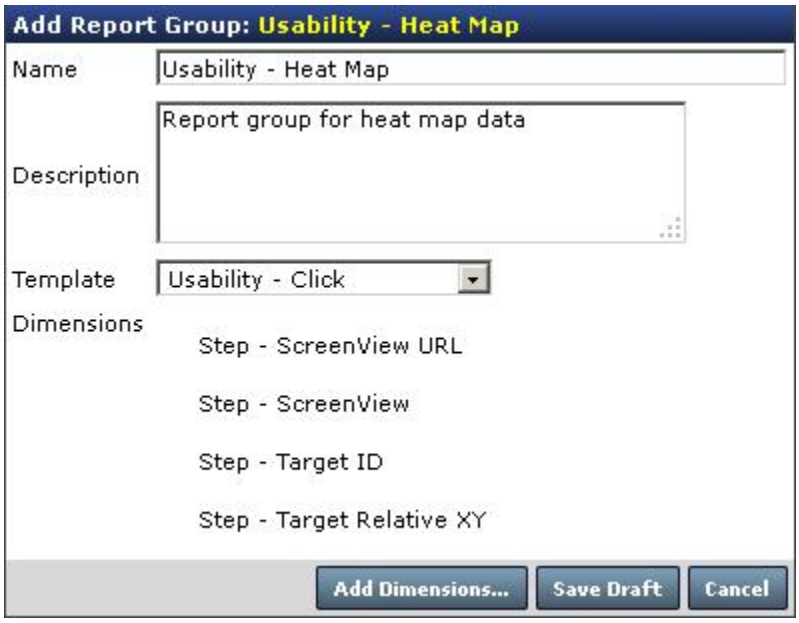

Figure 6. Creating report groups with a Usability template

The following report group templates are available to capture for IBM Tealeaf cxOverstat:

Note: A report group that is created from one of the IBM Tealeaf cxOverstat report group templates can contain up to eight dimensions.

Note: After you select a report group template, choosing a new report group template automatically removes all dimensions from the report group and starts with the included dimensions of the newly selected report group template.

Table 13. cxOverstat Report Group Templates	
Name	**Description**
`Usability - Click`	Events operating on comparison analytics, heat map, or link analytics data require these contextual dimensions
`Usability - Attention Map`	Events operating on attention map data require these contextual dimensions
`Usability - Form Analytics`	Events operating on form analytics data require these contextual dimensions

See "TEM Dimensions Tab" in the *IBM Tealeaf Event Manager Manual*.

Tracking other usability events

The IBM Tealeaf UI Capture solution provides more usability data that is not natively captured by Tealeaf. Specifically, you can create custom step messages to submit data of interest to you pertaining to the client activities of your web application.

See "Step-Based Eventing" in the *IBM Tealeaf Event Manager Manual*.

Default Tealeaf client framework event objects

Tealeaf provides default events and event-related objects for detecting, capturing, and storing data from the Tealeaf client framework(s) that you deploy in your web environment. As you build your object library, you might want to print this page and complete the gaps with the names of the objects you created. You can also gather this information at any time by reviewing the dependent objects for any selected object in the Event Manager.

Generated user agent attributes

This table lists and describes the attributes that are generated by Tealeaf based on user agent information.

Hit Attribute	Event	Dimension	Session Attribute	Description
"Tealeaf Standard Event Object Reference	"Tealeaf Standard Event Object Reference	"Tealeaf Standard Event Object Reference		Detection of source of hit from mobile native application
"Tealeaf Standard Event Object Reference" (=`MOBILE`)		"Tealeaf Standard Event Object Reference		Detection of Mobile Web hit captured by IBM Tealeaf UI Capture
"Tealeaf Standard Event Object Reference (=`MOBILE_APP`)		"Tealeaf Standard Event Object Reference		Detection of Mobile Web hit captured by a Tealeaf mobile client framework

Objects for XML version of UI Capture for Ajax

This table lists and describes the objects that are provided to support detection of data that is captured by XML from the IBM Tealeaf UI Capture solution.

Note: These objects were provided by Tealeaf to support the legacy XML version of IBM Tealeaf UI Capture, in which client events were submitted in XML format. In the future, only the JSON version of IBM Tealeaf CX UI Capture for AJAX will be supported, and these objects will be deprecated.

Hit Attribute	Event	Dimension	Session Attribute	Description
"Tealeaf Standard Event Object Reference"				Count of JavaScript alerts for the hit, as reported by IBM Tealeaf CX UI Capture for AJAX
"Tealeaf Standard Event Object Reference"				The application name, as reported by IBM Tealeaf CX UI Capture for AJAX
"Tealeaf Standard Event Object Reference"				The resolution of the browser, as reported by IBM Tealeaf CX UI Capture for AJAX
"Tealeaf Standard Event Object Reference"				The count of client user interface events for the hit, as reported by IBM Tealeaf CX UI Capture for AJAX
"Tealeaf Standard Event Object Reference"				The size of the event message for the hit in characters, as reported by IBM Tealeaf CX UI Capture for AJAX
"Tealeaf Standard Event Object Reference"				Hit count, as reported by IBM Tealeaf CX UI Capture for AJAX
"Tealeaf Standard Event Object Reference"			"Tealeaf Standard Event Object Reference"	This event is populated by a system session attribute that monitors the running total of CUI hits for the current session.
"Tealeaf Standard Event Object Reference"				Dwell time in milliseconds for the hit, as reported by IBM Tealeaf CX UI Capture for AJAX

Hit Attribute	Event	Dimension	Session Attribute	Description
"Tealeaf Standard Event Object Reference"				The count of images that failed to load for the hit, as reported by IBM Tealeaf CX UI Capture for AJAX
"Tealeaf Standard Event Object Reference"				Render time in milliseconds for the hit, as reported by IBM Tealeaf CX UI Capture for AJAX
"Tealeaf Standard Event Object Reference"				Type of client user interface hit, as reported by IBM Tealeaf CX UI Capture for AJAX
"Tealeaf Standard Event Object Reference"				The unnormalized URL for the client hit, as reported by IBM Tealeaf CX UI Capture for AJAX
"Tealeaf Standard Event Object Reference"				The normalized URL for the client hit, as reported by IBM Tealeaf CX UI Capture for AJAX

Legacy objects

This table lists and describes the objects that are maintained in Tealeaf to support legacy versions of the Tealeaf client frameworks, including IBM Tealeaf CX UI Capture for AJAX.

Avoid using these objects, as they are likely to be deprecated in a future release.

Table 14. Legacy objects

Hit Attribute	Event	Dimension	Session Attribute	Description
"Tealeaf Standard Event Object Reference" in the *IBM Tealeaf Event Manager Manual*				(Legacy) Detection of hit from mobile native application **Note:** This hit attribute is used only for legacy versions of the Tealeaf Logging Frameworks. Do not use.

Table 14. Legacy objects (continued)				
Hit Attribute	**Event**	**Dimension**	**Session Attribute**	**Description**
Client UI Hit (T/F)				(Legacy) Detection of hit from IBM Tealeaf CX UI Capture for AJAX legacy versions **Note:** This hit attribute is used only for legacy versions of the IBM Tealeaf CX UI Capture for AJAX solution. Do not use.

Tealeaf JSON object schema reference

Tealeaf's client frameworks support a standardized messaging system for submitting user interface events and user actions from the client to Tealeaf for capture. This JSON-based messaging system provides an efficient means of processing these data streams on the client and within Tealeaf.

During the client capture process, Tealeaf formats captured events into JSON structures that are easily processed by Tealeaf. The section provides information about the JSON schema that is used by Tealeaf client frameworks, including format, features, and any differences in the supported features between individual client frameworks.

This feature affects the clients generating logging data to a standardized format. The current clients are the following:

- "UI Capture j2 Guide" in the *IBM Tealeaf UI Capture for j2 Guide*
- "UI Capture for Ajax Guide" in the *IBM Tealeaf UI Capture for Ajax Guide*
- "Tealeaf Android Logging Framework Reference Guide" in the *IBM Tealeaf Android Logging Framework Reference Guide*
- "Tealeaf iOS Logging Framework Reference Guide" in the *IBM Tealeaf iOS Logging Framework Reference Guide*

Table of Contents

Design features

The JSON schema applies to Release 8.5 and later versions of the Tealeaf client frameworks.

Extensible

The data format supports new data sources and new features in existing sources when they appear in the capture stream. The data interchange format can be modified to accommodate the new data. This data normalization simplifies the event and step attribute creation process.

Normalized interactions

The data format normalizes captured interactions across all client frameworks. For example, the JSON message that indicates when the visitor changed a form field value has the same structure for each of the supported client frameworks.

Compatibility with an earlier version

By design, the schema enables the capture across multiple versions of the client frameworks. The data in the JSON schema is not compatible with an earlier version with Release 8.4 or earlier. Version number information is included whenever client environmental data is submitted.

Ease of use

To economize on space and network bandwidth, JSON messages are bundled together and submitted in a compact structure. In the native frameworks, the request payload is also compressed in compressed format.

When the data is received by Tealeaf, it is decompressed by the PCA and inserted into the request of the parent hit with which it is associated.

When you replay the session in Browser-Based Replay, the request of each hit that contains JSON messages includes a reformatted easier-to-read version of the JSON messages.

I18N/L10N

UTF-8 is the supported string format because this is the format of the request buffer.

Other generalized features

JSON is the format of choice because:

- It is a native format for JavaScript engines. The Tealeaf event engine is one of them.
- It is quickly replacing XML as a de-facto standard in RIA.
- Latest browser versions have native support for JSON.

Unified header format

Identifying client framework sessions

Each client framework submits data to Tealeaf for capturing by using a consistent header format. The format is in the following structure:

```
X-Tealeaf: device (platform) Lib/libversion
```

This header identifies the client framework and the version of the framework from which the hit was submitted. Below, you can see examples from each of the supported client frameworks:

```
X-Tealeaf: device (Android) Lib/0.0.7
X-Tealeaf: device (iOS) Lib/0.1.3
X-Tealeaf: device (UIC) Lib/2008.3.6.1
```

In the PCA pipeline, this header is detected and rendered as a request variable. This is an example of one of the request variables:

```
HTTP_X_TEALEAF=device (Android) Lib/0.0.7
```

Content-encoding and content-type headers

To reduce network traffic, the native client logging frameworks are compressing the request data into compressed format before submitting them to Tealeaf for capture. For each submitted hit, its `Content-Encoding` header is set to the value `gzip`.

Before PCA Build 3502, you must configure the IBM Tealeaf CX Passive Capture Application to capture `Content-Encoding=*/gzip` type.
Beginning in PCA Build 3502, this content encoding type is automatically captured for you.

Additionally, each client framework submits hits by using this content-type header:

```
Content-Type=application/json
```

If you deployed a IBM Tealeaf CX UI Capture for AJAX solution that is configured to submit user interface data through XML, this header has different values.

Session identifiers

Depending on the type of client framework, the identifier for the session information varies.

Consider this example:

```
[StepAttributes]
{
    "serialNumber": 1,
    "messageVersion": "0.0.0.2",
    "sessions": [
        {
            "startTime": 1331346014596,
            "id": "C4B290A5E3EB4E28FA8CE6DCCDDE6CF5",

...
```

For the `id` value:

- In IBM Tealeaf Android SDK, the `id` value identifies the session. It is created automatically, or the application user passes a unique ID that is used for sessionization. This value is the same for each hit of a single session.
- In IBM Tealeaf iOS SDK, the `id` value identifies the session. It is created automatically, or the application user passes a unique ID that is used for sessionization. This value is the same for each hit of a single session.

  This data is not submitted by IBM Tealeaf CX UI Capture for AJAX.

The `id` value for sessions is not a consistent value for identifying sessions. Use `TLSID`, which is inserted into the request of each parent hit of a set of step attributes.

Count steps

Each step includes a `count` property, which identifies the number of the step for the current hit.

In the example below, you can see the `count` properties in each of the two steps for this hit: `type=LOAD` and `type=UNLOAD`. For the `count` properties, values start at `1` and are automatically incremented.

```
"messages": [
        {
            "type": 2,
            "offset": 0,
            "screenviewOffset": 0,
            "count": 1,
            "fromWeb": true,
            "screenview": {
                "type": "LOAD",
                "name": "root",
                "url": "/",
                "referrer": ""
            }
        },

        {
            "type": 2,
```

```
            "offset": 40824,
            "screenviewOffset": 0,
            "count": 12,
            "fromWeb": true,
            "screenview": {
                "type": "UNLOAD",
                "name": "root",
                "url": "/",
                "referrer": ""
            }
        }
```

Performance measurement

Tealeaf has the ability to capture individual hits from the client enables a unique ability to monitor the performance of your web application, the network, and the client browser.

Connection type messages

In the submitted JSON messages, connection type information is submitted as message `type=3`.

The connection type message is not sent by IBM Tealeaf CX UI Capture for AJAX.

For example:

```
{
    "offset": 0,
    "type": 3,
    "connection": {
        "statusCode": 200,
        "responseDataSize": 0,
        "initTime": 0,
        "responseTime": 0,
        "url": "http://www.example.com",
        "loadTime": 0
    }
}
```

From the iOS Logging Framework, some of the properties are not available for capture.

Time offsets from page load and start of session

In framework messages, time events are measured based on the page load for each page.

```
[StepAttributes]
{
    "serialNumber": 1,
    "messageVersion": "0.0.0.2",
    "sessions": [
        {
            "startTime": 1331346014596,
            "id": "C4B290A5E3EB4E28FA8CE6DCCDDE6CF5",

...
```

In this example, the value for `startTime` indicates the time offset in milliseconds since Jan 1, 1970 UTC when the page was loaded from which the set of attributes was generated.

Each individual attribute has a time value in the `offset` property:

```
"messages": [
        {
            "type": 2,
            "offset": 0,
            "screenviewOffset": 0,
            "count": 1,
            "fromWeb": true,
```

```
            "screenview": {
                "type": "LOAD",
                "name": "root",
                "url": "/",
                "referrer": ""
            }
        },
        {
            "type": 2,
            "offset": 40824,
            "screenviewOffset": 0,
            "count": 12,
            "fromWeb": true,
            "screenview": {
                "type": "UNLOAD",
                "name": "root",
                "url": "/",
                "referrer": ""
            }
        }
```

In these example messages, the `offset` property contains two different values:

- For the first message, it indicates that the LOAD event occurred 0 milliseconds after page load.
- For the second message, it indicates that the UNLOAD event occurred 40824 milliseconds after page load, approximately 41 seconds after the page has loaded.

renderTime offsets

In the first message, the `renderTime` property indicates that the time taken to render the page required 254 milliseconds, which should match the value for the offset property for the message.

In IBM Tealeaf UI Capture, the `renderTime` property is reported in the Performance message in milliseconds.

Dwell time

Dwell time is applicable for control types where the visitor can spend time within the control. For example, when the visitor clicks in a text input control, the dwell time measures the time in milliseconds between when the control receives focus and when the focus switches to a different control.

- Controls that do not support a visitor spending time in them do not contain the dwell time property.
- From IBM Tealeaf UI Capture, dwell time is not provided for change events that are not associated with a blur event. For example, if a user clicks several times in a row on a check box, no dwell time value is provided.

In this example, the `dwell` property measures this value in milliseconds for a textbox.

```
{
    "type": 4,
    "offset": 8465,
    "count": 8,
    "event": {
        "type": "change"
    },
    "target": {
        "id": "billing:lastname",
        "idType": -1,
        "type": "INPUT",
        "subType": "text",
        "name": "billing[lastname]",
        "dwell": 5620,
        "currState": {
            "value": "asf"
        }
    }
}
```

Link image targets

UI Capture j2 can capture the content of the `href` target element if the element:

- Is a link
- Has a parent that is a link (`isParentLink=true`).

In this example, the `href` value records the link destination. In this case, the link is listed for the parent object, as `isParentLink` is set to `true`.

```
{
    "type": 4,
    "offset": 13456,
    "screenviewOffset": 13455,
    "count": 9,
    "fromWeb": true,
    "target": {
        "id": "logo",
        "idType": -1,
        "name": "",
        "tlType": "IMG",
        "type": "IMG",
        "subType": "",
        "position": {
            "width": 18,
            "height": 20,
            "relXY": "0.6,0.5"
        },
        "currState": {
            "href": "http://example.com/mypage.html#logo_link"
        },
        "isParentLink": true,
        "visitedCount": 1
    },
    "event": {
        "tlEvent": "click",
        "type": "click"
    },
    "focusInOffset": 3207
},
```

Previous state and current state tracking

Each control object submits information to identify its previous state before the change to the current state.

If a field is empty, no value is submitted for it. As a result, no value for `currState` is recorded.

```
"target": {
    "position": {
        "y": 0,
        "height": 100,
        "width": 200,
        "x": 0
    },
    "id": "nameTextBox",
    "dwell": 232313,
        "noFilled": 2,
    "type": "MyEditText",
    "prevState": {
         "text": ""
    },
    "currState": {
                 "text": "MyName"
    },
    "subType": "EditText",
    "tlType": "textBox"
},"screenViewOffset": 1253,"offset": 67676876,"screenViewOffset": 4556,"type":
    4,"event": {
    "type": "click",
    "subType": "subclick"
}
```

Exceptions

The Tealeaf client frameworks capture exceptions that are recorded in the client application and submits them in a specific message type (`type=6`).

Mobile native client frameworks

When a stack trace is detected on the client application, a JSON message of `type=6` is submitted, including the entire stack trace of the exception in the `stackTrace` property, as in this divide-by-zero example:

```
{"offset": 0,"type": 6,"exception": {
    "description": "divide by zero",
    "stackTrace": "java.lang.ArithmeticException: divide by zero\n\tat
com.tl.uic.test.model.JSONTest.testException(JSONTest.java:391)\n\tat
java.lang.reflect.Method.invokeNative(Native Method)\n\tat
java.lang.reflect.Method.invoke(Method.java:507)\n\tat
android.test.InstrumentationTestCase.runMethod(InstrumentationTestCase.java:204
)\n\tat
android.test.InstrumentationTestCase.runTest(InstrumentationTestCase.java:194)\
n\tat
android.test.ActivityInstrumentationTestCase2.runTest(ActivityInstrumentationTe
stCase2.java:186)\n\tat
junit.framework.TestCase.runBare(TestCase.java:127)\n\tat
junit.framework.TestResult$1.protect(TestResult.java:106)\n\tat
junit.framework.TestResult.runProtected(TestResult.java:124)\n\tat
junit.framework.TestResult.run(TestResult.java:109)\n\tat
junit.framework.TestCase.run(TestCase.java:118)\n\tat
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:169)\n\tat
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:154)\n\tat
android.test.InstrumentationTestRunner.onStart(InstrumentationTestRunner.java:5
29)\n\tat
android.app.Instrumentation$InstrumentationThread.run(Instrumentation.java:1448
)\n",
    "name": "java.lang.ArithmeticException"
}
```

Some stack trace messages cannot be captured from the iOS Logging Framework.

Form field monitoring

Tealeaf client frameworks provide superior insight into specific activities on the form fields of each page of your web application. Once per hit, the Tealeaf client framework submits information about the form fields that were left blank or were unchanged, as well as information about which fields were changed multiple times. You can also monitor how long the visitor spent in an individual form field.

Message on form fields that were unchanged

To monitor form fields that were not changed, the Tealeaf client frameworks submit a message of `type=4`. This message is for a synthetic event of `type=unchanged`. This synthetic event message is submitted whenever the control value for an object on logical page unload is the same value as detected during logical page load.

Reported along with this information is the set of form fields on the hit that were not changed, as described in the `target` set of properties in this example:

```
{
    "type": 4,
    "offset": 5469,
    "count": 7,
    "event": {
        "type": "unchanged"
    },
    "target": {
        "id": "cb1",
        "idType": -1,
        "type": "INPUT",
        "name": "cb1",
        "subType": "checkbox",
```

```
        "visitedCount": 0,
        "currState": {
            "checked": false,
            "value": "red"
        }
    }
}
```

This message is submitted when an ScreenView is unloaded, which might not correspond to when the page is unloaded. If the ScreenView can be unloaded multiple times per page in your application, the unchanged message might be reported multiple times, which might affect reporting.

Show fields completed more than once

Tealeaf client frameworks can also monitor the form fields that were visited multiple times on a single page. In this example, the `visitedCount` property is inserted into the target message, and its value (2) indicates the number of times the form field (`id=nameTextBox`) was visited:

```
{"target": {
    "position": {
        "y": 0,
        "height": 100,
        "width": 200,
        "x": 0
    },
    "id": "nameTextBox",
    "dwell": 232313,
    "visitedCount": 2,
    "type": "MyEditText",
    "prevState": {
        "text": ""
    },
    "currState": {
         "text": "MyName"
    } ,
    "subType": "EditText",
    "tlType": "textBox"
},"offset": 67676876,"screenViewOffset": 4556,"type": 4,"event": {
    "type": "click",
    "subType": "subclick"
}
}
```

Message on form fields that were left blank

For each form field control that was not entered, the Tealeaf client frameworks submit a message that indicate that the form field was left blank.

- This message is submitted when the user selected the field but did not enter any value.
- The `visitedCount` for this message is a value greater than `0`.

ScreenView features

A ScreenView is defined as a change in state on the page. For example, on a web page that uses tabs to identify steps in a process, each tab can be defined as a screenview.

What is a screenview

The definition of a screenview is specified by the application developer. The Tealeaf client frameworks submit events that are triggered based on changes to the hash value in the URL. These events are categorized as the hashchange event.

Each client framework records the screenview message when it occurs. A screenview can be triggered by the application

1. Explicitly starting an API that notifies the Tealeaf library that a new screenview is loaded.

2. Triggering a hashchange on the page URL.

In this case, the Tealeaf library must be configured to track hashchange events and to use them to infer screenview load.

This is an example screenview message from a checkout page:

```
{
    "type": 2,
    "offset": 4672,
    "screenviewOffset": 0,
    "count": 5,
    "fromWeb": true,
    "screenview": {
        "type": "LOAD",
        "name": "billing",
        "url": "/index.php/checkout/onepage/",
        "referrer": "CheckoutMethod"
    }
}
```

When the ScreenView changes, the client reset the count of steps and assume that the content displayed in the client application was changed to display a new view or screen.

LOAD type

The LOAD type is structured as follows:

```
{
    "type": 2,
    "offset": 0,
    "screenviewOffset": 0,
    "count": 1,
    "fromWeb": true,
    "screenview": {
        "type": "LOAD",
        "name": "root",
        "url": "/",
        "referrer": ""
    }
}
```

Among the properties, these values are present for the LOAD message:

- `type` value is LOAD.
- `name` value is the logical page name.
- `url` value is the relative path from the root for the page URL.
- `referrer` value is the referrer to the current page.

UNLOAD type

The UNLOAD type is structured as follows:

```
{
                    "type": 2,
                    "offset": 40824,
                    "screenviewOffset": 0,
                    "count": 12,
                    "fromWeb": true,
                    "screenview": {
                        "type": "UNLOAD",
                        "name": "root",
                        "url": "/",
                        "referrer": ""
                    }
                }
```

Among the properties, these values are present for the UNLOAD message:

- `type` value is UNLOAD.
- `name` value is the logical page name.
- `url` value is the relative path from the root for the page URL.
- `referrer` value is the referrer to the current page.

Track objects unchanged within a given screenView

When the ScreenView changes, the client frameworks submit a message containing the list of objects on the page that were not changed from their initial values.

JSON data message format

For compactness of the format, fields are stored as array elements, and optional fields are stored in objects. A message packet consists of a global header followed by one or more message types. Each message packet contains a global header that consists of an array that contains the message version and a message serial number. The header is followed by an array of session messages.

There are two types of messages:

- Session
- Client environment

Message format schema

This example of the JSON message format schema:

```
{
    "messageVersion": {
        "title": "Json message version",
        "type": "string",
        "required": true
    },
    "serialNumber": {
        "title": "Number of packet of MessageFormat sent to server",
        "type": "integer",
        "required": true
    },
    "clientEnvironment": {
        "title": "Client environment",
        "type": "$ref": "ClientEnvironment",
        "required": true
    },
    "sessions": {
        "description": "We simplified this to an array of one session.",
        "type": "array",
        "additionalItems": {
            {
                "description": "Session with list of messages",
                "type": "array",
                "item": {
                    { "$ref" : "Session" },
                    "required": true,
                }
            },
        },
    },
    "additionalProperties" : false
    }
}
```

Message format example

This example contains a message header and the client environment information that is submitted by the client framework from an Android native application:

```
{
    "serialNumber": 0,
    "messageVersion": "0.0.0.1",
    "sessions": [
        {
            "startTime": 1328311295574,
            "id": "945202AC4E93104E05EDADE1F6059B97",
            "messages": [
                {
                    "offset": 124,
                    "screenViewOffset": 4556,
                    "type": 2,
                    "logicalPageName": "HomeActivity"
                }
            ]
        }
      ],
    "clientEnvironment": {
        "mobileEnvironment": {
            "android": {
                "keyboardType": "QWERTY",
                "brand": "generic",
                "fingerPrint":       "generic/sdk/generic/
                :2.2/FRF91/43546:eng/test-keys"
            },
            "totalMemory": 63422464,
            "totalStorage": 12288,
            "orientationType": "PORTRAIT",
            "appVersion": "1.0.5",
            "manufacturer": "unknown",
            "userId": "android-build",
            "locale": "English (United States)",
            "deviceModel": "sdk",
            "language": "English"
        },
        "width": 0,
        "height": 0,
        "osVersion": "2.2"
    }
}
```

Message format

Each message packet contains a global header consisting of an array containing the message version and a message serial number. The header is followed by an array of session messages.

Session messages

The Session messages in a JSON packet contain message type objects to be posted to server.

Session messages schema

This is the schema for Session messages.

```
{
    "id": {
        "title": "Session id",
        "type": "string",
        "required": true
    },
    "startTime": {
        "title": "Session start time in ticks",
        "type": "number",
        "required": true
    },
    "messages": {
        "description": "List of messages",
        "type": "array",
        "additionalItems": {
            {
```

```
                "description": "List of messages",
                "type": "array",
                "item": {
                    { "$ref" : "Client State" },
                    { "$ref" : "ScreenView" },
                    { "$ref" : "Connection" },
                    { "$ref" : "Control" },
                    { "$ref" : "Custom Event" },
                    { "$ref" : "Exception" },
                    "required": true,
                }
            },
        },
    },
   "additionalProperties" : false
}
```

Session messages example
This example shows a Session message.

```
{
    "startTime": 1328311295574,
    "id": "945202AC4E93104E05EDADE1F6059B97",
    "messages": [
        {
            "offset": 124,
            "screenViewOffset": 4556,
            "type": 2,
            "logicalPageName": "HomeActivity"
        }
    ]
}
```

Client environment data
Client environment data contains session-level information that is sent during all the messages sent to the server. It is part of the message format. IBM TealeafCX UI Capture for AJAX does not send `clientEnvironment` messages.

Client environment schema
This is the client environment message schema.

```
{
    "$ref" : "MessageHeader",
    "osVersion": {
        "title": "Operating system version",
        "type": "string",
        "required": false for UIC, true for native frameworks
    },
   "orientation": {
        "title": "Initial session orientation of the screen",
        "type": "integer which can be 0, 90, 180 or -90",
        "required": true for iOS and Android. UIC has an orientation
in webEnvironment.
    },
    "height": {
        "title": "Initial session height of display/viewport divided
by pixel density",
        "type": "integer",
        "required": true
    },
    "width": {
        "title": "Initial session width of display/viewport divided
by pixel density",
        "type": "integer",
        "required": true
    },
   "deviceHeight": {
        "title": "Initial session device height of display/viewport",
        "type": "integer",
        "required": true
    },
```

```
    "deviceWidth": {
        "title": "Initial session device width of display/viewport",
        "type": "integer",
        "required": true
    },
    "mobileEnvironment": {
        "description": "Logical page being loaded",
        "type": "object",
        "properties": {
            "appName": {
                "title": "Application name",
                "type": "string",
                "required": true
            },
            "android": {
                "description": "Current state in an Android device",
                "type": "object",
                "properties": {
                    "brand": {
                        "title": "The brand (e.g., carrier) the software
is customized for, if any",
                        "type": "string",
                        "required": true
                    },
                    "fingerPrint": {
                        "title": "A string that uniquely identifies this build",
                        "type": "string",
                        "required": true
                    },
                    "keyboardType": {
                        "title": "Keyboard type",
                        "type": [ {
                            "enum": [0],
                            description: "TWELVE_KEYS"
                            },
                            "enum": [1],
                            description: "NO_KEYS"
                            },
                            "enum": [2],
                            description: "QWERTY"
                            },
                            "enum": [3],
                            description: "UNDEFINED"
                            }],
                        "required": true
                    },
                }
            },
            "totalMemory": {
                "title": "Total memory in MBytes of device",
                "type": "number",
                "required": true
            },
            "totalStorage": {
                "title": "Total storage in MBytes of device",
                "type": "number",
                "required": true
            },
            "orientationType": {
                "title": "Orientation type of device",
                "type": [ {
                            "LANDSCAPE"
                            description: "LANDSCAPE"
                            },
                            "PORTRAIT",
                            description: "PORTRAIT"
                            },
                            "SQUARE",
                            description: "SQUARE"
                            },
                            "UNDEFINED",
                            description: "UNDEFINED"
                            }],
                "required": true
            },
            "appVersion": {
                "title": "Version of application",
                "type": "string",
                "required": true
            },
            "manufacturer": {
                "title": "Manufacturer of device",
```

```
                "type": "string",
                "required": true
            },
            "userId": {
                "title": "User of device",
                "type": "string",
                "required": true
            },
            "locale": {
                "title": "The user's preferred locale",
                "type": "string",
                "required": true
            },
            "deviceModel": {
                "title": "Device model",
                "type": "string",
                "required": true
            },
            "language": {
                "title": "Device's language",
                "type": "string",
                "required": true
            }
        },
        "additionalProperties" : false
    },
    “webEnvironment": {
        "description": "Web page being loaded",
        "type": "object",
        "properties": {
            "libVersion": {
                "title": "Library version,
                "type": "string",
                "required": true for UIC
            },
            "screen": {
                "description": "Display of the web content",
                "type": "object",
                "properties": {
                    "orientation": {
                        "title": "Initial session orientation of the screen",
                        "type": "integer which can be 0, 90, 180 or -90",
                        "required": true for UIC. iOS and Android has it on
outer object.
                    },
                    "orientationMode": {
                        "title": "To indicate orientation mode.”,
                        "type": "string which can be Portrait or Landscape",
                        "required": true for UIC
                    }
                }
            },
            "page": {
                "title": "Url of the page",
                "type": "string",
                "required": true for UIC
            },

            "referrer": {

                "title": "Referrer URL (if any) of the page.",
                "type": "string",
                "required": true for UIC

            }
        },
        "additionalProperties" : false
    }
}
```

Client environment example

These examples contain a client environment messages that are submitted from applications.

This example shows the clientEnvironment message for UIC:

```
clientEnvironment" : {
    "webEnvironment" : {
```

```
        "libVersion" : "5.0.0.XXXX",
        "page" : "http://uictest.com/frames/",
        "referrer" : "http://uictest.com/",
        "screen" : {
            "devicePixelRatio" : 1,
            "deviceWidth" : 1920,
            "deviceHeight" : 1080,
            "deviceToolbarHeight" : 34,
            "width" : 942,
            "height" : 955,
            "orientation" : 0,
            "orientationMode" : "PORTRAIT"
        }
    }

}
```

This example shows the clientEnvironment message for Android and iOS:

```
"clientEnvironment": {
                "orientation": 90,
                "height": 720,
                "osVersion": "4.2.2",
                "pixelDensity": 2,
                "width": 1196,
                "deviceHeight": 360,
                "osType": "Android" or "iOS"
                "deviceWidth": 598
}
```

JSON message type schemas and examples

JSON messages are categorized by type for processing. Tealeaf supports 12 JSON message types.

Message header properties

All messages contain message header properties consisting of two properties that contain the message type and the time that is offset from the start of the session in milliseconds. All time measurements in the JSON object schema are in milliseconds.

Message list

This table lists and describes the supported JSON message types:

Table 15. Schema by Message Type

Type	Message Type	Description
1	**“Client state (Type 1) messages” on page 80**	Any object that shows the current state of client.
2	**“ScreenView (Type 2) messages” on page 83**	Any message that indicates changes in view on the "screen". The "screen" is the page, view, or activity where the visitor is in the application.
3	**“Connections (Type 3) messages” on page 84**	Any request or response that the application performs during capture.
4	**“Control (Type 4) messages” on page 85**	User interface control that fires an event to which Tealeaf listens for capture.
5	**“Custom Event (Type 5) messages” on page 88**	Any custom log event from any location in application.
6	**“Exception (Type 6) messages” on page 88**	Any exception that the application can throw.

Table 15. Schema by Message Type (continued)		
Type	**Message Type**	**Description**
7	**“Performance (Type 7) messages” on page 90**	Performance data from a browser.
8	**“Web Storage (Type 8) messages” on page 91**	Any object that contains information about local storage information on the browser.
9	**“Overstat Hover Event (Type 9) messages” on page 91**	Any object that contains information about mouse hover and hover-to-click activity.
10	**“Layout (Type 10) messages” on page 92**	Any message that shows the current display layout of a native page.
11	**“Gesture (Type 11) messages” on page 94**	Any message that shows a gesture that fires a higher touch event that Tealeaf listens to for capture.
12	**“DOM Capture (Type 12) message example” on page 103**	Any object that contains serialized HTML data (DOM snapshot) of the page.
13	**“GeoLocation (Type 13) messages” on page 105**	Messages that contain the geolocation information about the device.

Message header properties
All messages contain message header properties consisting of two properties that contain the message type and the time that is offset from the start of the session in milliseconds.

All time measurements in the JSON object schema are in milliseconds.

Message header properties schema
This example shows the schema for the JSON message headers.

```
"offset": {
    "title": "Milliseconds offset from start of stream",
    "type": "integer",
    "required": true
},"screenViewOffset": {
    "title": "Milliseconds offset from start of ScreenView",
    "type": "integer",
    "required": true
},"count": {
    "title": "The number of the message being sent",
    "type": "integer",
    "required": only used for UIC
},"fromWeb": {
    "title": "Used to identify if it came from Web or Native application",
    "type": "boolean",
    "required": true
},"webviewId": {
    "title": "Used to identify which webview it came from. This is only used

          when fromWeb is true and it is a hybrid application ",
    "type":"string",
    "required": true only when fromWeb is true and it is a hybrid application
},"type": {
    "title": "Message header type",
    "type": [ {
        "enum": [1],
        description: "CLIENT_STATE"
        },
        "enum": [2],
        description: "APPLICATION_CONTEXT"
        }],
        "enum": [3],
        description: "CONNECTION"
        },
        "enum": [4],
        description: "CONTROL"
```

```
        },
        "enum": [5],
        description: "CUSTOM_EVENT"
        }],
        "enum": [6],
        description: "EXCEPTION"
        }],
    "required": true
},
```

Message header properties example

The following example message header is taken from an exception message type.

```
{
    "offset": 0,
    "screenViewOffset": 4556,
    "type": 6,
    "exception": {
        "description": "divide by zero",
        "name": "class java.lang.ArithmeticException"
      "stackTrace": "java.lang.ArithmeticException: divide by zero\n\tat
com.tl.uic.test.model.JSONTest.testException(JSONTest.java:391)\n\tat
java.lang.reflect.Method.invokeNative(Native Method)\n\tat
java.lang.reflect.Method.invoke(Method.java:507)\n\tat
android.test.InstrumentationTestCase.runMethod(InstrumentationTestCase.java:204
)\n\tat
android.test.InstrumentationTestCase.runTest(InstrumentationTestCase.java:194)\
n\tat
android.test.ActivityInstrumentationTestCase2.runTest(ActivityInstrumentationTe
stCase2.java:186)\n\tat
junit.framework.TestCase.runBare(TestCase.java:127)\n\tat
junit.framework.TestResult$1.protect(TestResult.java:106)\n\tat
junit.framework.TestResult.runProtected(TestResult.java:124)\n\tat
junit.framework.TestResult.run(TestResult.java:109)\n\tat
junit.framework.TestCase.run(TestCase.java:118)\n\tat
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:169)\n\tat
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:154)\n\tat
android.test.InstrumentationTestRunner.onStart(InstrumentationTestRunner.java:5
29)\n\tat
android.app.Instrumentation$InstrumentationThread.run(Instrumentation.java:1448
)\n",
    }
}
```

Client state (Type 1) messages

Client state messages are delivered on a schedule basis or on changes to the environment state on the client. These are Type 1 JSON messages.

Note: Replay of client state messages is not supported, except for scroll events. Replay of scroll events that are captured from the client is supported for mobile sessions (web, mobile web and hybrid) only in BBR only. Replay of scroll events are not supported for DOM capture. See *Search and Replay for Mobile Web*.

Client State (Type 1) message schema

This is the schema for the Client State (Type 1) messages.

```
{
    "$ref" : "MessageHeader",
    "mobileState": {
        "description": "Logical page being loaded for iOS and Android",
        "type": "object",
        "properties": {
            "orientation": {
                "title": "Current orientation of the device",
                "type": "integer",
                "required": true
            },
            "freeStorage": {
                "title": "Amount of available storage in Mbytes",
                "type": "number",
```

```
            "required": true
        },
        "androidState": {
            "description": "Current state in an Android device",
            "type": "object",
            "properties": {
                "keyboardState": {
                    "title": "Current keyboard state",
                    "type": [ {
                        "enum": [0],
                        description: "Keyboard not hidden"
                        },
                        "enum": [1],
                        description: "Keyboard hidden"
                        },
                        "enum": [2],
                        description: "Undefined"
                        }],
                    "required": true
                },
            }
        },
        "battery": {
            "title": "Battery level from 0 to 100",
            "type": "number",
            "required": true
        },
        "freeMemory": {
            "title": "Amount of available memory in Mbytes",
            "type": "number",
            "required": true
        },
        "connectionType": {
            "title": "Current connection type",
            "type": "string",
            "required": true
        },
        "carrier": {
            "title": "Carrier of device",
            "type": "string",
            "required": true
        },
        "networkReachability": {
            "title": "Current network reachability",
            "type": [ {
                "enum": [0],
                description: "Unknown"
                },
                "enum": [1],
                description: "NotReachable"
                },
                "enum": [2],
                description: "ReachableViaWIFI"
                },
                "enum": [3],
                description: "ReachableViaWWAN"
            }],
            "required": true
        },
        "ip": {
            "title": "Ip address of device",
            "type": "string",
            "required": true
        }
    },
    "additionalProperties" : false
    "clientState": {
    "description": "Logical web page being loaded for UIC",
    "type": "object",
    "properties": {
        "pageWidth": {
            "title": "Width of the document of the web page",
            "type": "integer",
            "required": true
        },
        "pageHeight": {
            "title": "Height of the document of the web page",
            "type": "integer",
            "required": true
        },
        "viewPortWidth": {
            "title": "Width of viewport",
```

```
                "type": "integer",
                "required": true
            },
            "viewPortHeight": {
                "title": "Height of viewport",
                "type": "integer",
                "required": true
            },
            "viewPortX": {
                "title": "x position of scrollbar on viewport",
                "type": "integer",
                "required": true
            },
            "viewPortY": {
                "title": "y position of scrollbar on viewport",
                "type": "integer",
                "required": true
            },
            "event": {
                "title": "event that triggered the client state",
                "type": "string",
                "required": true
            },
                    "deviceScale": {
                        "title": "scaling factor for fitting
                        page into window for replay",
                        "type": "integer",
                        "required": true
                    },
                    "viewTime": {
                "title": "time in milliseconds user was on the event triggered",
                "type": "integer",
                "required": true
            },
            "viewPortXStart": {
                "title": "initial start x position of scrollbar on viewport",
                "type": "integer",
                "required": only used in scroll events
            },
            "viewPortYStart": {
                "title": "initial start y position of scrollbar on viewport",
                "type": "integer",
                "required": only used in scroll events
            },
        },
        "additionalProperties" : false
    }
}
```

Client State (Type 1) message example

This is an example of a Client State (Type 1) message. This example comes from an Android native application.

```
{
    "offset": 667,
    "screenViewOffset": 4556,
    "type": 1,
    "mobileState": {
        "orientation": 0,
        "freeStorage": 33972224,
        "androidState": {
            "keyboardState": 0
        },
        "battery": 50,
        "freeMemory": 64630784,
        "connectionType": "UMTS",
        "carrier": "Android",
        "networkReachability": "ReachableViaWWAN",
        "ip": "0.0.0.0"
    }
}
```

ScreenView (Type 2) messages

ScreenView messages indicate steps in a visitor's experience with your application. These steps can be logical page views in a web application, screen changes in a mobile application, or steps in a business process. ScreenView messages are Type 2 JSON messages.

In Release 8.5 and earlier, these messages were called Application Context messages.

ScreenView (Type 2) message schema

This is the schema for the ScreenView (Type 2) JSON messages.

```
{
    "$ref" : "MessageHeader",
    "dcid": {
        "title": "Unique identifier that is used to match the corresponding
DOM Capture message associated with this
message.",
        "type": "string",
        "required": false
    },
    "screenview/context": {
        "description": "Logical page being loaded or unloaded",
        "type": "object",
        "properties": {
            "type": {
                "title": "Type of application context - LOAD or UNLOAD",
                "type": "string",
                "required": true
            },
            "name": {
                "title": "Name of the logical page. This is given by customer
or it uses name of the class used
by the page.",
                "type": "string",
                "required": true
            },
            "url": {
                "title": "URL path of the logical page",
                "type": "string",
                "required": false only used in UIC
            },
            "host": {
                "title": "URL Host of the logical page",
                "type": "string",
                "required": false only used in UIC
            },
            "referrer": {
                "title": "Previous logical page loaded, only used in LOAD",
                "type": "string",
                "required": false
            },
            "referrerUrl": {
                "title": "Url of the previous logical page loaded",
                "type": "string",
                "required": false, not used in UIC
            }
        },
        "additionalProperties" : false,
        "required": false
    }
}
```

ScreenView (Type 2) message example

This is an example of a ScreenView (Type 2) message. This example contains three ScreenView messages, indicating page load and page unload events.

```
{
    "offset": 124,
    "contextOffset": 4556,
    "type": 2,
    "context": {
            "type": "LOAD",
            "name": "PAGE 2",
            "referrer": "PAGE 1"
        }
```

```
}

{
    "type": 2,
    "offset": 19216

    "context": {
        "type": "UNLOAD",
        "name": "PAGE 2"
    }
}

{
    "type": 2,
    "offset": 2144,
    "contextOffset": 0,
    "count": 9,
    "fromWeb": true,

    "webviewId": "webview1",
    "screenview": {
        "type": "LOAD",
        "name": "Ford",
        "url": "/dynamic/ford.aspx",

        "host": "http://www.cartest.com",
        "referrer": "BMW",
        "referrerUrl": "/dynamic/bmw.aspx"
    }
}
```

Connections (Type 3) messages

Connection messages provide information about how requests or responses are managed by the client application. Connections messages are Type 3 JSON messages.

Connections (Type 3) messages schema

This is the schema for Connections (Type 3) JSON messages.

```
{
    "$ref" : "MessageHeader",
    "connection": {
        "description": "Connection in application",
        "type": "object",
        "properties": {
            "statusCode": {
                "title": "Status code of connection",
                "type": "integer",
                "required": true
            },
            "responseDataSize": {
                "title": "Response data size",
                "type": "number",
                "required": true
            },
            "initTime": {
                "title": "Initial time of connection",
                "type": "number",
                "required": true
            },
            "responseTime": {
                "title": "Response time of connection",
                "type": "number",
                "required": true
            },
            "url": {
                "title": "Url of connection",
                "type": "string",
                "required": true
            },
            "loadTime": {
                "title": "Load time from connection",
                "type": "number",
                "required": true
            }
        },
        "additionalProperties" : false
```

```
    }
}
```

Connections (Type 3) message example
This example shows the Connections (Type 3) JSON message.

```
{
     "offset": 03829,
        "type": 3,
        "screenViewOffset": 45560,
    "type": 3,
    "connection": {
         "statusCode": 200,
             "responseDataSize": 0272,
         "initTime": 01333669478556,
             "responseTime": 02237,
             "url": "http://google.com",
             "url": "/store/js/tealeaf/
                    TeaLeafTarget.php??width=540&height=960&orientation=0",
             "loadTime": 0
    }
}
```

Control (Type 4) messages
Control messages are used to log user action and behavior. These messages consist of a control identifier and a value that is returned by the identified control. Control messages are Type 4 JSON messages.

The control identifiers are mapped to specific controls for the submitting client framework. The value can be a number, a text string, or structured data.

Control (Type 4) message schema
This is the schema for Control (Type 4) messages.

The X and Y properties are not present in the UI Capture frameworks.

```
{
    "$ref" : "MessageHeader",
      "offset": {
        "title": "Milliseconds offset from offset
                      for when focusIn of text fields occur",
        "type": "integer",
        "required": true
       },
    "target": {
        "description": "Control being logged",
        "type": "object",
        "properties": {
            "position": {
                "description": "Position of control being logged",
                "type": "object",
                "properties": {
                    "x": {
                        "title": "X of the control",
                        "type": "integer",
                        "required": true
                    },
                    "y": {
                        "title": "Y of the control",
                        "type": "integer",
                        "required": true
                    },
                    "height": {
                        "title": "height of control",
                        "type": "integer",
                        "required": true
                    },
                    "width": {
                        "title": "width of control",
                        "type": "integer",
                        "required": true
                    },
```

```
                    "relXY": {
                        "title": "relative X & Y ratio that
                                                can be from 0 to 1 with a
                                                default value of 0.5",
                        "type": "string",
                        "required": true for click events
                    },
                },
                "additionalProperties" : false
            }
            "id": {
                "title": "Id/Name/Tag of control",
                "type": "string",
                "required": true
            },
            idType": {
                "title": "Indicates what id is based on: Native id (e.g. HTML 'id'
attribute):  -1,
    xPath: -2, or Custom attribute for UIC and
Hashcode value for  Native: -3, or xPath for Native iOS/Android: -4",
                "type": "integer",
                "required": true
            },
            "dwell": {
                "title": "Dwell time of control",
                "type": "integer value that is in milliseconds",
                "required": false
            },
             "visitedCount": {
                "title": "Number of times a form control has
                                been visited to be filled  by user.",
                "type": "integer",
                "required": false
            },
            "isParentLink": {
                "title": "To indicate if control a A type tag",
                "type": "boolean",
                "required": false only in UIC for usability
            },
            "name": {
                "title": "Name of control",
                "type": "string",
                "required": true in UIC
            },
            "type": {
                "title": "Type of control",
                "type": "string",
                "required": true
            },
            "subType": {
                "title": "SubType of control",
                "type": "string",
                "required": true
            },
                    "tlType": {
                "title": "tlType of control that normalizes
                                the control type for eventing",
                "type": "string",
                "required": true
            },
                    "prevState": {
                "title": "Previous state of control",
                "type": "object",
                "required": true,
                                "properties": {
                    "?": { // Could be any variable name given by developer
                        "title": "Additional data in string format",
                        "type": "string",
                        "required": false
                    }
            },
            "currState": {
                "title": "Current state of control",
                "type": "object",
                "required": true,
                "properties": {
                    "?": { // Could be any variable name given by developer
                        "title": "Additional data in string format",
                        "type": "string",
                        "required": false
                    }
            }
```

```
        },
        "additionalProperties" : false
    }
    "event": {
        "description": "Event from control",
        "type": "object",
        "properties": {
                   "tlEvent": {
                "title": "Tealeaf type of event",
                "type": "string",
                "required": true
            },
            "type": {
                "title": "Type of event",
                "type": "string",
                "required": true
            },
            "subType": {
                "title": "Subtype of event",
                "type": "string",
                "required": true
            }
        },
        "additionalProperties" : false
    }
}
```

Control (Type 4) message example

This is an example of a Control (Type 4) message.

This example shows a control with an `idType` of XPATH, which means no `id` was assigned to the control in the application so Tealeaf traversed the layout and created an XPATH `id` for the control:,

```
{
    "screenviewOffset":380,
    "target":{
        "id":"[KV,0]",
        "position":{
            "y":331,
            "x":0,
            "width":320,
            "height":202
            },
        "idType":"-4",
            "currState":{
                "y":"0",
                "x":"0"
                },
        "style":{
            "paddingTop":2,
            "textBGAlphaColor":255,
            "bgAlphaColor":255,
            "paddingBottom":0,
            "paddingLeft":0,
            "hidden":false,
            "paddingRight":0
            },
        "subType":"View",
        "type":"KeyboardView",
        "tlType":"keyboard"
       },
    "type":4,
    "offset":728,
    "count":3,
    "fromWeb":false,
    "event":{
        "type":"UIKeyboardDidShowNotification",
        "tlEvent": "kbDisplayed"
        }
},
```

Custom Event (Type 5) messages

The Custom Event messages are used to custom log any event from any place in the application. Custom Event messages are Type 5 JSON messages.

Custom Event (Type 5) message schema

This is the schema for the Custom Event (Type 5) messages.

The only required field is the name of the custom event (`name` value). Application-specific code must be created to process this logged message type.

```
{
    "$ref" : "MessageHeader",
    "customEvent": {
        "description": "Custom event message",
        "type": "object",
        "properties": {
            "name": {
                "title": "Exception name/type",
                "type": "string",
                "required": true
            },
                    "data": "Additional properties given by developer",
                "type": "object",
                "required": truefalse,
                "properties": {
                    "?": { // Could be any variable name given by developer
                        "title": "Additional data in string format",
                        "type": "string",
                        "required": false
                    }
                        },
        },
        "additionalProperties" : false
    }
}
```

Custom Event (Type 5) message example

This is an example of a Custom Event (Type 5) message. This custom event message provides the name of the custom event (`MyEvent_1`) and several custom properties in the `data` section.

```
{
     "type": 5,
        "offset": 17981,
    "screenViewOffset": 4556,
    "customEvent": {
        "name": "MyEvent_1",
        "data": {
            "Foo": "Bar",
            "validationError": "Invalid zipcode.",
            "ajaxPerformance": 56734
        }
    }
}
```

Exception (Type 6) messages

The exceptions messages type records the name and description of an exception occurring on the client application. Exception messages are Type 6 JSON messages.

Exception (Type 6) message schema

This is the schema for the Exception (Type 6) messages.

```
{
    "$ref" : "MessageHeader",
    "exception": {
        "description": "Exception description message",
        "type": "object",
        "properties": {
```

```
            "description": {
                "title": "Exception message from api call",
                "type": "string",
                "required": true
            },
            "name": {
                "title": "Exception name/type",
                "type": "string",
                "required": true, not for UIC
            },
            "stackTrace": {
                "title": "Exception stacktrace given by framework",
                "type": "string",
                "required": true, not for UIC
            },
            "url": {
                "title": "Url where exception ocurred",
                "type": "string",
                "required": true for UIC
            },
           "fileName": {
                "title": "File name where exception ocurred",
                "type": "string",
                "required": true for iOS, not for UIC
            },
            "line": {
                "title": "Line number where eception occurred.",
                "type": "string",
                "required": true for UIC and iOS
            },
           "unhandled": {
                "title": "Whether exception had a try catch around it.",
                "type": "boolean",
                "required": true, not for UIC
            },
           "data": {
                "title": "User defined data being passed with
user info from system",
                "type": "object",
                "required": true for iOS, not for UIC
    "properties": {

                "userInfo": {
                            "type": "object",

    "title": "OS information from error or exception",

                            "required": iOS optional (data is JSON
serializable or not)
                        },
                        "message": {
                            "type": "string",

                            "title":"User supplied message on error event",

                            "required":iOS optional (not on exceptions
required on error)
                        }
                    },
            },
        },
        "additionalProperties" : false
    }
}
```

Exception (Type 6) message example

This is an example of an Exception (Type 6) message. This example exception indicates an attempt to read a property named 'type' of a variable or value which is undefined.

```
{
                    "type" : 6,
                    "offset" : 4606,
                    "screenviewOffset" : 4603,
                    "count" : 3,
                    "fromWeb" : true,
                    "exception" : {
                        "description" : "Uncaught TypeError: Cannot read
```

```
property 'type' of undefined",
                        "url" : "http://www.xyz.com/js/badscript.js",
                        "line" : 258
                    }
}
```

Performance (Type 7) messages

Performance messages show performance data from a browser. Performance messages are Type 7 JSON messages.

Performance (Type 7) message schema

This is the schema for Performance (Type 7) messages.

```
{
    "$ref" : "MessageHeader",
    "performance": {
        "description": "Performance message",
        "type": "object",
        "properties": {
                    },
        "additionalProperties" : false
    }
}
```

Performance (Type 7) message example

This is an example of a Performance (Type 7) message.

```
{
    "type": 7,
    "offset": 9182,
    "screenviewOffset": 9181,
    "count": 3,
    "fromWeb": true,
    "performance": {
        "timing": {
             "redirectEnd": 0,
             "secureConnectionStart": 0,
             "domainLookupStart": 159,
             "domContentLoadedEventStart": 2531,
             "domainLookupEnd": 159,
             "domContentLoadedEventEnd": 2551,
             "fetchStart": 159,
             "connectEnd": 166,
             "responseEnd": 1774,
             "domComplete": 2760,
             "responseStart": 728,
             "requestStart": 166,
             "redirectStart": 0,
             "unloadEventEnd": 0,
             "domInteractive": 2531,
             "connectStart": 165,
             "unloadEventStart": 0,
             "domLoading": 1769,
             "loadEventStart": 2760,
             "navigationStart": 0,
             "loadEventEnd": 2780,
             "renderTime": 986
            },
         "navigation": {
             "type": "NAVIGATE",
             "redirectCount": 0
        }
    }
}
```

Web Storage (Type 8) messages

Web Storage messages are any objects that contain information about local storage information on the browser. Web Storage messages are Type 8 JSON messages.

Web Storage (Type 8) message schema

This is the schema for the Web Storage (Type 8) messages.

```
"$ref" : "MessageHeader",
webStorage: {
    key : &ldquo;string&rdquo;,
    value: &ldquo;string&rdquo;,
}
```

Web Storage (Type 8) message example

This is an example of a Web Storage (Type 8) message.

```
{
    type: 8,
    offset: 25,
    screenviewOffset: 23,
    count: 2,
    fromWeb: true,
    webStorage: {
        key: "vistCount"
        value: "5"
    }
}
```

Overstat Hover Event (Type 9) messages

Overstat Hover Event messages are any object containing information about mouse hover and hover-to-click activity. Overstat Hover Event messages are Type 9 JSON messages.

Overstat Hover Event (Type 9) message schema

This is the schema for Overstat Hover Event (Type 9) messages

```
"$ref" : "MessageHeader",
event: {
    xPath: "string",
    hoverDuration: int,
    hoverToClick: boolean,
    gridPosition: {
        x: int,
        y: int
     }
}
```

Overstat Hover Event (Type 9) message example

This is an example of a Overstat Hover Event (Type 9) message.

```
{
    type: 9,
    offset: 25,
    screenviewOffset: 23,
    count: 2,
    fromWeb: true,
    event: {
        xPath: "[\"ii\"]",
        hoverDuration: 5457,
        hoverToClick: false,
        gridPosition: {
            x: 3,
            y: 2
        }
}
```

Layout (Type 10) messages

Layout messages show the current display layout of a native page. Layout messages are Type 10 JSON messages.

Layout (Type 10) message schema

This is the schema for Layout (Type 10) messages.

```
"$ref" : "MessageHeader",
"version": {
  "description": "Message Version, must be in x.x format",
  "type": "string",
  "required": true
                },
"layoutControl": {
  "description": "Control on application page",
  "type": "object",
  "properties": {
    "position": {
      "description": "Position of control",
      "type": "object",
      "properties": {
        "x": {
          "title": "X of the control",
          "type": "integer",
          "required": true
        },
        "y": {
          "title": "Y of the control",
          "type": "integer",
          "required": true
        },
        "height": {
          "title": "height of control",
          "type": "integer",
          "required": true
        },
        "width": {
          "title": "width of control",
          "type": "integer",
          "required": true
        }
      },
      "additionalProperties" : false
    }
    "id": {
      "title": "Id/Name/Tag of control",
      "type": "string",
      "required": true
    },
    "type": {
      "title": "Type of control",
      "type": "string",
      "required": true
    },
    "subType": {
      "title": "SubType of control",
      "type": "string",
      "required": true
    },
    "tlType": {
      "title": "tlType of control that normalizes the control type for eventing",
      "type": "string",
      "required": true
    },
    "currState": {
      "title": "Current state of control",
      "type": "object",
      "required": true,
      "properties": {
        "?": { // Could be any variable name given by developer
            "title": "Additional data in string format",
            "type": "string",
            "required": false
        }
      }
    },
    "style" : {
      "title": "Style of the control",
```

```
      "type": "object",
      "required": true,
      "properties": {
        "textColor": {
          "title": "Text color",
          "type": "string",
          "required": true
        },
        "textAlphaColor": {
          "title": "Text alpha color",
          "type": "string",
          "required": true
        },
        "textBGColor": {
          "title": "Text background color",
          "type": "string",
          "required": true
        },
        "textBGAlphaColor": {
          "title": "Text background alpha color",
          "type": "string",
          "required": true
        },
        "bgColor": {
            "title": "Background color",
            "type": "string",
            "required": true
        },
        "bgAlphaColor": {
            "title": "Background alpha color",
            "type": "string",
            "required": true
        }
      }
    }
  },
  "additionalProperties" : false
}
```

Layout (Type 10) message example

This is an example of a Layout (Type 10) message.

```
{
    "offset": 27004,
    "screenviewOffset": 4706,
    "count": 16,
    "fromWeb": false,
    "type": 10,
    "version" : "1.0",
    "orientation" : 0,
    "deviceHeight": 592,
    "deviceWidth":  360,
    "layout": {
        "name": "loginPage",
        "class": "loginPageActivty",
        "controls": [
            {
                "position": {
                    "y": 38,
                    "height": 96,
                    "width": 720,
                    "x": 0
                },
                "id": "com.tl.uiwidget:id\/userNameLabel",
                "idType": -1,
                "type": "UILabel",
                "subType": "UIView",
                "tlType": "label",
                "currState": {
                    "text": "User name*"
                },
                "style": {
                    "textColor": 16777215,
                    "textAlphaColor": 1,
                    "textBGColor": 0,
                    "textBGAlphaColor": 0,
                    "bgColor": 0,
                    "bgAlphaColor": 0
                }
```

```
            },
            {...},
            {...}
        ]
    }
}
```

Gesture (Type 11) messages

Gesture messages are used to log user action and behavior. A Gesture message consists of a control identifier and a the value returned by that control. The control identifiers are mapped to specific controls on the client logging platform. The value can be a number, a text string or structured data. Gesture messages are Type 12 JSON messages.

Gesture (Type 11) message schema

This is the schema for Gesture (Type 11) messages.

Touch events

This is a JSON object that represents a gesture finger that is linked to control underneath the finger. It is reused in targets property of gesture type 11.This is the schema for touch events:

```
{
      "$ref" : "MessageHeader",
      "focusInOffset": {
      "title": "Milliseconds offset from offset for when focusIn of text
fields occur",
      "type": "integer",
      "required": false
      },
      "target": {
      "description": "Control being logged",
      "type": "object",
      "properties": {
          "position": {
              "description": "Position of control being logged",
              "type": "object",
              "properties": {
                  "x": {
                      "title": "X of the control",
                      "type": "integer",
                      "required": true
                  },
                  "y": {
                      "title": "Y of the control",
                      "type": "integer",
                      "required": true
                  },
                  "height": {
                      "title": "height of control",
                      "type": "integer",
                      "required": true
                  },
                  "width": {
                      "title": "width of control",
                      "type": "integer",
                      "required": true
                  },
                  "relXY": {
                      "title": "relative X & Y ratio that can be
                                from 0
to 1 with a default value of 0.5",
                      "type": "string",
                      "required": true for click events
                  },
                  "scrollX": {
                     "title": "scroll X of the page",
                     "type": "integer",
                     "required": true
                  },
                  "scrollY": {
                     "title": "scroll Y of the page",
                      "type": "integer",
                      "required": true
                  },
              },
```

```
            "additionalProperties" : false
        }
        "id": {
            "title": "Id/Name/Tag of control",
            "type": "string",
            "required": true
        },
        "idType":{
            "title": "Indicates what id is based on: Native id (e.g. HTML
'id' attribute): -1,
             xPath: -2, or Custom attribute for
UIC and Hashcode value for Native: -3, or xPath for Native iOS/Android: -4",
             "type": "integer",
             "required": true
        },
        "dwell": {
            "title": "Dwell time of control",
            "type": "integer value that is in milliseconds",
            "required": false
        },
        "focusInOffset": {
            "title": "Offset when control got focus",
            "type": "integer value that is in milliseconds",
            "required": true in UIC
        },
        "visitedCount": {
            "title": "Number of times a form control has been visited to be
filled by
      user.",
            "type": "integer",
            "required": false
        },
        "isParentLink": {
            "title": "To indicate if control a A type tag",
            "type": "boolean",
            "required": false only in UIC for usability
        },
        "name": {
            "title": "Name of control",
            "type": "string",
            "required": true in UIC
        },
        "type": {
            "title": "Type of control",
            "type": "string",
            "required": true
        },
        "subType": {
            "title": "SubType of control",
            "type": "string",
            "required": true
        },
        "tlType": {
            "title": "tlType of control that normalizes the control type for
     eventing",
            "type": "string",
            "required": true
        },
        "prevState": {
            "title": "Previous state of control",
            "type": "object",
            "required": false,
            "properties": {
                "?": { // Could be any variable name given by developer
                    "title": "Additional data in string format",
                    "type": "string",
                    "required": false
                }
            }
        },
        "currState": {
            "title": "Current state of control",
            "type": "object",
            "required": true,
            "properties": {
                "?": { // Could be any variable name given by developer
                    "title": "Additional data in string format",
                    "type": "string",
                    "required": false
                }
            }
        }
```

```
      },
      "additionalProperties" : false
      }
      "event": {
      "description": "Event from control",
      "type": "object",
      "properties": {
          "tlEvent": {
              "title": "Tealeaf type of event",
              "type": "string",
              "required": true
          },
          "type": {
              "title": "Type of event",
              "type": "string",
              "required": false
          },
          "subType": {
              "title": "Subtype of event",
              "type": "string",
              "required": false
          }
      },
      "additionalProperties" : false
      }}
```

Tap event schema

This contains only one touch object. This is the schema for tap events:

```
{
    "$ref" : "MessageHeader",
    "event": {
        "description": "Event from control",
        "type": "object",
        "properties": {
            "tlEvent": {
                "title": "Tealeaf type of event",
                "type": "string",
                "required": true
            },
            "type": {
                "title": "Type of event framework reports",
                "type": "string",
                "required": false
            }
        }
    },
    "touches": {
        "description": "Gestures touch objects per finger.",
        "type": "array",
        "required": true
        "items": {
                "description": "Touch objects per finger starting with intial
and ends with last object when finger is lifted from device.",
                "type": "array",
                "required": true,
                "$ref": "Touch"
            }
        }
    }
}
```

Swipe event schema

The swipe event contains only one touch object which will be the initial location with its corresponding direction and velocity. This is the schema for swipe events:

```
{
    "$ref" : "MessageHeader",
    "event": {
        "description": "Event from control",
        "type": "object",
        "properties": {
            "tlEvent": {
                "title": "Tealeaf type of event",
```

```
                "type": "string",
                "required": true
            },
            "type": {
                "title": "Type of event framework reports",
                "type": "string",
                "required": false
            }
        }
    },
    "touches": {
        "description": "Gestures touch objects per finger.",
        "type": "array",
        "required": true
        "items": {
                "description": "Touch objects per finger starting with intial
and ends with last object when finger is lifted from device.",
                "type": "array",
                "required": true,
                "$ref": "Touch"
            }
        }
    },
    "direction": {
        "title": "The direction of the swipe which can be up, down. left or
right.",
        "type": "string",
        "required": true
    },
    "velocityX": {
        "title": "The velocity of this measured in pixels per second along the
x axis",
        "type": "float",
        "required": true
    },
    "velocityY": {
        "title": "The velocity of this measured in pixels per second along the
y axis",
        "type": "float",
        "required": false
    }
}
```

Pinch events

The pinch event contains only an initial touch object per finger and the last touch object per finger, with the corresponding direction. This is the schema for pinch events:

```
{
    "$ref" : "MessageHeader",
    "event": {
        "description": "Event from control",
        "type": "object",
        "properties": {
            "tlEvent": {
                "title": "Tealeaf type of event",
                "type": "string",
                "required": true
            },
            "type": {
                "title": "Type of event framework reports",
                "type": "string",
                "required": false
            }
        }
    },
    "touches": {
        "description": "Gestures touch objects per finger.",
        "type": "array",
        "required": true
        "items": {
                "description": "Touch objects per finger starting with intial and
ends with last object when finger is lifted from device.",
                "type": "array",
                "required": true,
                "$ref": "Touch"
            }
        }
    },
```

```
    "direction": {
        "title": "Direction of pinch which can be open or close",
        "type": "string",
        "required": true
    }
}
```

Gesture (Type 11) message example

These are examples of UIC SDK Gesture (Type 11) messages.

Tap events

This example is a gesture message for a tap event:

```
{
  "fromWeb": false,
  "type": 11,
  "offset": 46788,
  "screenviewOffset": 42208,
  "count": 14,
  "event": {
    "type": "ACTION_DOWN",
    "tlEvent": "tap"
  },
  "touches": [
    [
      {
        "position": {
          "x": 179,
          "y": 543
        },
        "control": {
          "position": {
            "height": 184,
            "width": 1080,
            "relXY": "0.17,0.93"
              "scrollX": 10
              "scrollY": 15
          },
          "id": "[RL,0]",
          "idType": -4,
          "type": "RelativeLayout",
          "subType": "ViewGroup",
          "tlType": "canvas"
        }
      }
    ]
  ]
}
```

Double tap events

This example is a gesture message for a double tap event:

```
{
  "fromWeb": false,
  "type": 11,
  "offset": 49585,
  "screenviewOffset": 45005,
  "count": 15,
  "event": {
    "type": "ACTION_DOWN",
    "tlEvent": "doubleTap"
  },
  "touches": [
    [
      {
        "position": {
          "x": 182,
          "y": 520
        },
        "control": {
          "position": {
            "height": 184,
            "width": 1080,
            "relXY": "0.17,0.8"
```

```
              "scrollX": 10
              "scrollY": 15
          },
          "id": "[RL,0]",
           "idType": -4,
          "type": "RelativeLayout",
          "subType": "ViewGroup",
          "tlType": "canvas"
        }
      }
    ]
  ]
}
```

Tap hold events

This example is a gesture message for a tap hold event:

```
{
  "fromWeb": false,
  "type": 11,
  "offset": 52389,
  "screenviewOffset": 47809,
  "count": 16,
  "event": {
    "type": "ACTION_DOWN",
    "tlEvent": "tapHold"
  },
  "touches": [
    [
      {
        "position": {
          "x": 182,
          "y": 536
        },
        "control": {
          "position": {
            "height": 184,
            "width": 1080,
            "relXY": "0.17,0.89"
              "scrollX": 10
              "scrollY": 15
          },
          "id": "[RL,0]",
          "idType": -4,
          "type": "RelativeLayout",
          "subType": "ViewGroup",
          "tlType": "canvas"
        }
      }
    ]
  ]
}
```

Swipe event example

The swipe event contains only one touch object which will be the initial location with its corresponding direction and velocity. This example is a message for a swipe event:

```
{
  "fromWeb": false,
  "type": 11,
  "offset": 54409,
  "screenviewOffset": 49829,
  "count": 17,
  "event": {
    "type": "ACTION_DOWN",
    "tlEvent": "swipe"
  },
  "direction": "right",
  "velocityX": 7762.8466796875,
  "velocityY": 127.47991943359375,
  "touches": [
    [
      {
        "position": {
```

```
          "x": 75,
          "y": 538
        },
        "control": {
          "position": {
            "height": 184,
            "width": 1080,
            "relXY": "0.07,0.9"
              "scrollX": 10
              "scrollY": 15
          },
          "id": "[RL,0]",
          "type": "RelativeLayout",
          "subType": "ViewGroup",
          "tlType": "canvas"
        }
      },
      {
        "position": {
          "x": 212,
          "y": 526
        },
        "control": {
          "position": {
            "height": 184,
            "width": 1080,
            "relXY": "0.2,0.84"
              "scrollX": 10
              "scrollY": 15
          },
          "id": "[RL,0]",
          "idType": -4,
          "type": "RelativeLayout",
          "subType": "ViewGroup",
          "tlType": "canvas"
        }
      }
    ]
  ]
}
```

Pinch events

The pinch event contains only an initial touch object per finger and the last touch object per finger, with the corresponding direction. This example is a message for a pinch event:

```
{
    "type": 11,
    "offset": 2220,
    "screenviewOffset": 2022,
    "count": 6,
    "fromWeb": false,
    "event": {
        "tlEvent": "pinch",
        "type": "onScale"
    },
    "touches": [
        [
            {
                "position": {
                    "y": 388,
                    "x": 0
                },
                "control": {
                    "position": {
                        "height": 100,
                        "width": 100,
                        "relXY": "0.6,0.8"
                          "scrollX": 10
                          "scrollY": 15
                    },
                    "id": "com.tl.uic.appDarkHolo:id/imageView1",
                    "idType": -1,
                    "type": "ImageView",
                    "subType": "View",
                    "tlType": "image"
                }
            },
            {
```

```
                "position": {
                    "y": 388,
                    "x": 400
                },
                "control": {
                    "position": {
                        "height": 100,
                        "width": 100,
                        "relXY": "0.4,0.7"
                          "scrollX": 10
                          "scrollY": 15
                    },
                    "id": "com.tl.uic.appDarkHolo:id/imageView1",
                    "idType": -1,
                    "type": "ImageView",
                    "subType": "View",
                    "tlType": "image"
                }
            }
        ],
        [
            {
                "position": {
                    "y": 388,
                    "x": 800
                },
                "control": {
                    "position": {
                        "height": 100,
                        "width": 100,
                        "relXY": "0.6,0.8"
                          "scrollX": 10
                          "scrollY": 15
                    },
                    "id": "com.tl.uic.appDarkHolo:id/imageView1",
                    "idType": -1,
                    "type": "ImageView",
                    "subType": "View",
                    "tlType": "image"
                }
            },
            {
                "position": {
                    "y": 388,
                    "x": 500
                },
                "control": {
                    "position": {
                        "height": 100,
                        "width": 100,
                        "relXY": "0.4,0.7"
                          "scrollX": 10
                          "scrollY": 15
                },
                    "id": "com.tl.uic.appDarkHolo:id/imageView1",
                    "idType": -1,
                    "type": "ImageView",
                    "subType": "View",
                    "tlType": "image"
                }
            }
        ]
    ],
   "direction": "close"
}
```

DOM Capture (Type 12) messages

DOM Capture messages are objects that contain serialized HTML data (DOM snapshot) of the page. DOM Capture Messages are Type 12 JSON messages.

DOM Capture (Type 12) message schema

This is the schema for the DOM Capture (Type 12) messages.

```
"$ref" : "MessageHeader",
"domCapture": {
    "description": "Serialized HTML snapshot of the document.",
    "type": "object",
    "properties": {
```

```
        "dcid": {
            "title": "Unique identifier of this DOM snapshot.",
            "type": "string",
            "required": true
        }
          "fullDOM": {
            "title": "Flag indicating if the contents of this message contain
a full DOM or a DOM diff.",
            "type": "boolean",
            "required": false
        },
            "charset": {
            "title": "Browser reported charset of the document.",
            "type": "string",
            "required": false
        },
        "root": {
            "title": "Serialized HTML of the document.",
            "type": "string",
            "required": false
        },
        "diffs": {
            "title": "List of DOM diff entries. Each entry can contain
a HTML Diff or an attribute diff.",
            "type": "array",
            "required": false,
            "Item": {
                "title": "An object containing the DOM diff. The diff can
be a HTML diff or an attribute diff.",
                "type": "object",
                "required": false,
                "properties": {
                    "xpath": {
                        "title": "The xpath of the node.",
                        "type": "string",
                        "required": true
                    },
                    "root": {
                        "title": "Serialized HTML of the node referred
by the xpath. Presence of this property constitutes a HTML diff.",
                        "type": "string",
                        "required": false
                    },
                    "attributes": {
                        "title": "List of attribute diff entries. Each entry
contains a single attribute diff corresponding to the node
referred by the xpath. Presence of this property constitutes
an attribute diff.",
                        "type": "array",
                        "required": false,
                        "Item": {
                            "title": "An object containing the attribute
diff.",
                            "type": "object",
                            "required": true,
                            "properties": {
                                "name": {
                                    "title": "The attribute name.",
                                    "type": "string",
                                    "required": true
                                },
                                "value": {
                                    "title": "The attribute value.",
                                    "type": "string",
                                    "required": true
                                }
                            }
                        }
                    }
                }
            }
        },
        "eventOn": {
            "title": "Flag indicating if Tealeaf eventing should be enabled
for this DOM Capture snapshot.",
            "type": "boolean",
            "required": false
        },
        "url": {
            "title": "URL path of the snapshot document",
            "type": "string",
            "required": false
```

```
        },
        "host": {
            "title": "URL Host of the snapshot document",
            "type": "string",
            "required": false
        },
        "error": {
            "title": "Error message",
            "type": "string",
            "required": false
        },
        "errorCode": {
            "title": "Error code corresponding to the error message.",
            "type": "integer",
            "required": false
        },
        "frames": {
            "title": "Serialized HTML of any child frames of the document",
            "type": "array",
            "required": false,
            "Item": {
                "title": "An object containing serialized HTML of the frame",
                "type": "object",
                "required": false,
                "properties": {

                    "tltid": {

                        "title": "Unique identifier for this frame. Same tltid is
added to the serialized HTML source of the parent."

                        "type": "string",
                        "required": true

                    },

                    "charset": {
                        "title": "Browser reported charset of the document.",
                        "type": "string",
                        "required": true
                    },
                    "url": {
                        "title": "URL path of the snapshot document",
                        "type": "string",
                        "required": true
                    },
                    "host": {
                        "title": "URL Host of the snapshot document",
                        "type": "string",
                        "required": true
                    },

                    "root": {
                        "title": "Serialized HTML of the document.",
                        "type": "string",
                        "required": true
                    }
                }
            }
        },
        "canvas" : {
            "title": "Serialized data of the canvas snapshot.",
            "type": "array",
            "required": false,
        }
    },
    "additionalProperties" : false
}
```

DOM Capture (Type 12) message example

This is an example of a DOM Capture (Type 12) message.

This example shows a DOM message with full DOM capture enabled:

```
{
    // DOM Capture messages use type 12
    "type": 12,
```

```
    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "fullDOM":true
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
                "url": "/h4/greeting.html"
            }
        ],
        "canvas": []
    }
}
```

This example shows a DOM capture message with DOM diff enabled:

```
{
    "type": 12,
    "offset": 13874,
    "screenviewOffset": 13861,
    "count": 6,
    "fromWeb": true,
    "domCapture": {
        "fullDOM": false,
        "diffs": [
            {
                "xpath": "[[\"html\",0],[\"body\",0],[\"div\",1]]",
                "root": "<div class=\"bluebg\"><div><div>Input 1<input
type=\"text\" name=\"ip-x-1\" value=\"\"></div></div></div>"
            }
        ],
        "dcid": "dcid-3.1437256358764",
        "eventOn": false
    }
}



DOM Diff (with HTML and attribute diff):

{
    "type": 12,
    "offset": 5794,
    "screenviewOffset": 5777,
    "count": 8,
    "fromWeb": true,
    "domCapture": {
        "fullDOM": false,
        "diffs": [
            {
                "xpath": "[[\"html\",0],[\"body\",0],[\"div\",2],[\"div\",1]]",
                "root": "<div>Select List:<select name=\"select.pvt\"><option
value=\"O1\" selected=\"selected\">1</option><option value=\"O2\">2</option>
<option value=\"O3\">3</option></select></div>"
            },
            {
                "xpath": "[[\"cb1\"]]",
                "attributes": [
                    {
                        "name": "style",
                        "value": "height: 13px; width: 13px; visibility: hidden;"
                    }
                ]
            },
            {
                "xpath": "[[\"container_1\"],[\"table\",0],[\"tbody\",0],
```

```
[\"tr\",2],[\"td\",1],[\"select\",0]]",
                "attributes": [
                    {
                        "name": "style",
                        "value": "visibility: hidden;"
                    }
                ]
            }
        ],
        "dcid": "dcid-3.1437256879815",
        "eventOn": false
    }

}
```

This example shows the error message when the captured DOM message length exceeds the configured threshold:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    // The DOM Capture data is namespaced in the domCapture object
    "domCapture": {
        // The "error" contains the verbose error message explaining why the
DOM Capture couldn't be performed.
        "error": "Captured length (18045) exceeded limit (10000).",

        // The "errorCode" contains the numeric code for this error message.
Currently, there is only 1 error message.
        "errorCode": 101,

        // The "dcid" property contains the unique string identifying this
DOM Capture within the page instance.
        "dcid": "dcid-1.1414088027401"
    }
}
```

GeoLocation (Type 13) messages

A GeoLocation message logs a user's location information. The message consists of a control identifier and a GeoLocation value. If the user has given the permission to use location data, GeoLocation returns latitude, longitude, accuracy values. If the user has not given permission to use location data, GeoLocation returns an error code and error string.

GeoLocation (Type 13) message schemas

This is the schema for the GeoLocation (Type 13) JSON messages:

This is the schema for messages from a device that the user has given permission to use location data:

```
"$ref" : "MessageHeader",
    "geolocation": {
      "lat": double,
      "long": double,
      "accuracy": float
      }
```

This is the schema for messages from a device that the user has not given permission to use location data:

```
"$ref" : "MessageHeader",
    "geolocation": {
      "errorCode": int,
      "error": "string",
      }
```

GeoLocation (Type 13) message examples

This is an example of the GeoLocation (Type 13) JSON message.

This is an example of a message from a device that the user has given permission to use location data:

```
{   "type": 13,
     "geolocation": {
     "lat": 37.5680,
     "long": -122.3292,
     "accuracy": 65
     }
}
```

This is an example of a message from a device that the user has not given permission to use location data:

```
{   "type": 13,
     "geolocation": {
      "errorCode": 201,
      "error": "permission denied",
    }
}
```

Differences between frameworks

There are differences in how the SDKs capture and support the generic schema.

Schema changes for UI Capture j2

In Release 8.6, Tealeaf introduces a new version of UI Capture. IBM Tealeaf UI Capture is designed to capture JSON messages that use the latest schema and to support the IBM Tealeaf cxOverstat product.

Release 8.6 and later requires IBM Tealeaf UI Capture to capture and transmit client-side user interface events.

Schema changes for UI Capture for Ajax

Differences for UI Capture for Ajax include:

1. UI Capture does not send connection type messages.
2. UI Capture does not send stack trace messages. Instead, URL, line number, and error message are submitted.
3. UI Capture does not automatically generate ScreenView `LOAD` and `UNLOAD` messages. Instead, the monitored web application must use the appropriate API to inform UI Capture to send them.
4. UI Capture does not submit `clientEnvironment` messages.

Schema changes for Tealeaf Android SDK

Differences for the Android SDK include:

1. The message property `count` is not supported.
2. The `timezoneOffset` property is not supported.
3. For message control type 4:
   - The position properties (x,y, height, width) are supported.
   - The property `idType` is not supported on Android.
   - The property `subType` is required.

iOS pageshow/pagehide normalization

iOS-based events to show and hide pages are normalized to the data standards of the JSON schema that is used by Tealeaf.

iOS event
Tealeaf JSON event

pageshow
load

pagehide
unload

Tealeaf JSON properties

Several JSON data objects and properties are used by the IBM Tealeaf UI Capture JavaScript library. The JSON format is a subset of the Tealeaf JSON schema object specification.

Locate information

When you search for information about a specific property, use the path information after the bracketed references as a search string for the rest of the page. For example, if you are interested in information about:

```
<global>.messageVersion
```

You can use the property as a search string for more reference information:

```
.messageVersion
```

High-level JSON structure

The high-level JSON data structure is:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 1,

    "sessions": [
      {
          "id": "ID12H15M50S521R0.25678676785555626",
          "startTime": 1324584950521,
          "timezoneOffset": 480,
          "messages": [ ... ],
          "clientEnvironment": {
              "webEnvironment": {
                  "libVersion":"1.0.0.152",
                  "page": "http://moksha.tealeaf.com/html4.html",
                  "windowId":"I7gF$t",
                  "screen":{
                      "orientation":0,
                      "orientationMode": "PORTRAIT"
                  }  // 'screen'
              }  // 'webEnvironment'
          }  // 'clientEnvironment'
      }  // anonymous session object
    ]  // 'sessions'
}  // anonymous global object
```

Object information

Each JSON property path description includes this information:

Identifier
Description

<global>
Refers to the anonymous global object that serves as a container for the JSON payload. For example,

```
<global>.messageVersion
```

<session>
Refers to the anonymous session object which in the "sessions" array. For example,

```
<session>.startTime
```

<message>
Refers to the anonymous message object within the "messages" array. The message object serves as a container for all supported message types.

charset
Browser reported charset of the document.

Supported frameworks: IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.domCapture.charset

Description
Browser reported charset of the document.

Value
Charset as reported by the browser. For a list of codes refer to: http://www.iana.org/assignments/character-sets/character-sets.xhtm

Limitations
None

Dependencies
Used by Replay to correctly interpret the serialized HTML.

Example

This example shows a Type 12 DOM Capture message:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
                "url": "/h4/greeting.html"
            }
        ],
        "canvas": []
    }
    }
}
```

clientState
The state of the client, including pageHeight, pageWidth, and so on.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.clientState`

Description
The state of the client, including pageHeight, pageWidth, and so on.

Value
Object. Refer to individual property descriptions for details.

Limitations
The clientState object is present only in Client State message types (`<global>.<session>.<message>.type = 1`)

Dependencies
None

Example

This example shows the Type 1 Client State message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 1,
                "offset": 7353,
                "screenviewOffset": 7352,
                "count": 2,
                "fromWeb": true,
                "clientState": {
                    "pageWidth": 954,
                    "pageHeight": 820,
                    "viewPortWidth": 877,
                    "viewPortHeight": 836,
                    "viewPortX": 0,
                    "viewPortY": 0,
                    "event": "load",
                    "viewTime": 0
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

count
The index of the message in the session. The count of the first message is 1. This field is a generic message property that is used in multiple message types.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.count`

Description
The index of the message in the session. The count of the first message is 1. This field is a generic message property that is used in multiple message types.

Value
Integer.

Limitations
None

Dependencies
Replay uses the count to order the UI events.

Examples

This example shows the count field in a DOM Capture (Type 12) message:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
                "url": "/h4/greeting.html"
            }
        ],
        "canvas": []
    }
    }
}
```

dcid
A unique identifier for the DOM Capture snapshot that helps in associating the type 12 DOM Capture with its corresponding type 4 or type 2 trigger message.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.domCapture.dcid`

Description
A unique identifier for the DOM Capture snapshot that helps in associating the type 12 DOM Capture with its corresponding type 4 or type 2 trigger message.

Value
An opaque string that is unique within the context of all other DOM snapshots that originate from this application page.

Limitations
None

Dependencies
Replay uses it to uniquely identify the DOM Snapshot that is associated with the corresponding type 4 and type 2 messages.

Examples

This example shows the dcid property in the Type 12 DOM Capture message:

```
{
    // DOM Capture messages use type 12
```

```
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
                "url": "/h4/greeting.html"
            }
        ],
        "canvas": []
    }
    }
}
```

description
A description of the exception.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.exception.description`

Description
A description of the exception.

Value
String

Limitations
None

Dependencies
None.

Example

This example shows a Type 6 Exception message with a description property:

```
{
                "type" : 6,
                "offset" : 4606,
                "screenviewOffset" : 4603,
                "count" : 3,
                "fromWeb" : true,
                "exception" : {
                    "description" : "Uncaught TypeError: Cannot read property
'type' of undefined",
                    "url" : "http://www.xyz.com/js/badscript.js",
                    "line" : 258
                }
}
```

deviceHeight
This is the initial session device height of the display/viewport in pixels.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.clientState.deviceHeight

Description
The initial session device height of the display/viewport in pixels.

Value
An integer.

Limitations
None

Dependencies
Used by replay

Example

This example shows the Client Environment message for UIC:

```
clientEnvironment" : {
    "webEnvironment" : {
        "libVersion" : "5.0.0.XXXX",
        "page" : "http://uictest.com/frames/",
        "referrer" : "http://uictest.com/",
        "screen" : {
            "devicePixelRatio" : 1,
            "deviceWidth" : 1920,
            "deviceHeight" : 1080,
            "deviceToolbarHeight" : 34,
            "width" : 942,
            "height" : 955,
            "orientation" : 0,
            "orientationMode" : "PORTRAIT"
        }
    }

}
```

This example shows the Client Environment message for Android and iOS:

```
"clientEnvironment": {
                "orientation": 90,
                "height": 720,
                "osVersion": "4.2.2",
                "pixelDensity": 2,
                "width": 1196,
                "deviceHeight": 360,
                "osType": "Android" or "iOS"
                "deviceWidth": 598
}
```

deviceWidth
This is the initial session device width of the display/viewport in pixels.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.clientState.deviceWidth

Description
The initial session device width of the display/viewport in pixels.

Value
An integer.

Limitations
None

Dependencies
Used by replay

Example

This example shows the Client Environment message for UIC:

```
clientEnvironment" : {
    "webEnvironment" : {
        "libVersion" : "5.0.0.XXXX",
        "page" : "http://uictest.com/frames/",
        "referrer" : "http://uictest.com/",
        "screen" : {
            "devicePixelRatio" : 1,
            "deviceWidth" : 1920,
            "deviceHeight" : 1080,
            "deviceToolbarHeight" : 34,
            "width" : 942,
            "height" : 955,
            "orientation" : 0,
            "orientationMode" : "PORTRAIT"
        }
    }

}
```

This example shows the Client Environment message for Android and iOS:

```
"clientEnvironment": {
                "orientation": 90,
                "height": 720,
                "osVersion": "4.2.2",
                "pixelDensity": 2,
                "width": 1196,
                "deviceHeight": 360,
                "osType": "Android" or "iOS"
                "deviceWidth": 598
}
```

error
Error text that explains why DOM Capture failed.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.domCapture.error`

Description
Error message that explains why DOM Capture failed.

Value
String

Limitations
None

Dependencies
None

Example

This example shows the Type 12 DOM Capture error message:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    // The DOM Capture data is namespaced in the domCapture object
```

```
    "domCapture": {
        // The "error" contains the verbose error message explaining why the
DOM Capture couldn't be performed.
        "error": "Captured length (18045) exceeded limit (10000).",

        // The "errorCode" contains the numeric code for this error message.
Currently, there is only 1 error message.
        "errorCode": 101,

        // The "dcid" property contains the unique string identifying this
DOM Capture within the page instance.
        "dcid": "dcid-1.1414088027401"
    }
}
```

errorCode
Error code corresponding to the error message.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.domCapture.errorCode`

Description
Error code corresponding to the error message.

Value
Number

Limitations
None

Dependencies
Used by Replay.

Example

This example shows the Type 12 DOM Capture error message:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    // The DOM Capture data is namespaced in the domCapture object
    "domCapture": {
        // The "error" contains the verbose error message explaining why the
DOM Capture couldn't be performed.
        "error": "Captured length (18045) exceeded limit (10000).",

        // The "errorCode" contains the numeric code for this error message.
Currently, there is only 1 error message.
        "errorCode": 101,

        // The "dcid" property contains the unique string identifying this
DOM Capture within the page instance.
        "dcid": "dcid-1.1414088027401"
    }
}
```

event (client state)

- Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.clientState.event`

Description

Value
Integer.

Limitations
None.

Dependencies
None

Example

This example shows a Type 1 Client state message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 1,
                "offset": 7353,
                "screenviewOffset": 7352,
                "count": 2,
                "fromWeb": true,
                "clientState": {
                    "pageWidth": 954,
                    "pageHeight": 820,
                    "viewPortWidth": 877,
                    "viewPortHeight": 836,
                    "viewPortX": 0,
                    "viewPortY": 0,
                    "event": "load",
                    "viewTime": 0
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

event (control)

- Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.event`

Description
An object containing information about the nature of the user action.

Value
Object. Refer to individual property descriptions for details.

Limitations
The event object is present only in control message types (`<global>.<session>.<message>.type = 4`)

Dependencies
Replay uses various members of the event object.

Example

This example shows a Type 4 Control message:

```
{
    "messageVersion": "1.1.0.0",
```

```
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {...},
                "event": {
                    "type": "click"
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

eventOn
Flag indicating whether Tealeaf eventing is enabled for this DOM Capture snapshot.

Supported frameworks: IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.domCapture.eventOn

Description
Flag indicating whether Tealeaf eventing is enabled for this DOM Capture snapshot. This flag is true for the first DOM capture of the page. For subsequent DOM captures of the page, the flag is false.

Value
True or False

Limitations
None

Dependencies
Used by Tealeaf Eventing to determine whether eventing is allowed for this snapshot.

Example

This example shows a Type 12 DOM capture message:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
                "url": "/h4/greeting.html"
            }
        ],
```

```
        "canvas": []
    }
}
```

<frames>.charset
Browser reported charset of the child frame of the document.

Supported frameworks: IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.domCapture.<frames>.charset

Description
Browser reported charset of the child frame of the document.

Value
Charset as reported by the browser. For a list of codes refer to: http://www.iana.org/assignments/character-sets/character-sets.xhtm

Limitations
None

Dependencies
Used by Replay to correctly interpret the serialized HTML.

Example

This example shows a Type 12 DOM Capture message with frames:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
                "url": "/h4/greeting.html"
            }
        ],
        "canvas": []
    }
    }
}
```

<frames>.host
URL Host of the child frame of the snapshot document.

Supported frameworks: IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.domCapture.<frames>host

Description
URL Host of the child frame of the snapshot document.

Value
URL host string.

Limitations
None

Dependencies
Used by Replay.

Example

This example shows a Type 12 DOM Capture message with frames:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
                "url": "/h4/greeting.html"
            }
        ],
        "canvas": []
    }
    }
}
```

<frames>.root
Serialized HTML of the child frame of the document.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.domCapture.<frames>.root`

Description
Serialized HTML of the child frame of the document.

Value

HTML

Limitations
None

Dependencies

Used by Tealeaf Eventing and Replay.

Example

This example shows a Type 12 DOM Capture message with frames:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
                "url": "/h4/greeting.html"
            }
        ],
        "canvas": []
    }
    }
}
```

<frames>.tltid

Unique identifier for this frame. Same `tltid` is added to the serialized HTML source of the parent.

Supported frameworks: IBM Tealeaf UI Capture

Path

`<global>.<session>.<message>.domCapture.<frames>.tltid`

Description

A unique identifier for this frame. Same `tltid` is added to the serialized HTML source of the parent.

Value

An opaque string that is unique within the context of this DOM snapshot message.

Limitations

None

Dependencies

Used by Replay.

Example

This example shows a Type 12 DOM capture message with frames:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
```

```
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
                "url": "/h4/greeting.html"
            }
        ],
        "canvas": []
    }
    }
}
```

<frames>.url
URL path of the child frame of the snapshot document.

Supported frameworks: IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.domCapture.<frames>.url

Description
URL path of the child frame of the snapshot document.

Value
URL path string.

Limitations
None

Dependencies
Used by Replay.

Example

This example shows a Type 12 DOM Capture message with frames:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
                "url": "/h4/greeting.html"
            }
        ],
        "canvas": []
    }
    }
}
```

fromWeb
A Boolean flag that indicates whether the message originated from a web browser or web view.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.fromWeb`

Description
A Boolean flag that indicates whether the message originated from a web browser or web view.

Value
True or false

Limitations
None

Dependencies
None

Example

This example shows a Type 4 Control message that came from a web view:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {...},
                "event": {...}
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

height
The height in CSS pixels of the target object.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.target.position.height`

Description
The height in CSS pixels of the target object.

Value
Integer

Limitations
None

Dependencies
cxOverstat uses this information for producing heat maps.

Example

This example shows a Type 4 Control message::

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {
                    "id": "bi",
                    "idType": -1,
                    "name": "buttonInput",
                    "type": "INPUT",
                    "subType": "button",
                    "position": {
                        "width": 67,
                        "height": 22,
                        "relXY": "0.5,0.3"
                    },
                    "prevState": {...},
                    "currState": {...},
                    "isParentLink": false
                },
                "event": {
                    "type": "click"
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

height (Client Environment)
The initial session height of the display/viewport divided by pixel density.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.clientState.height`

Description
The initial session height of the display/viewport divided by pixel density.

Value
An integer.

Limitations
None

Dependencies
Used by replay

This example shows the Client Environment message for UIC:

```
clientEnvironment" : {
    "webEnvironment" : {
        "libVersion" : "5.0.0.XXXX",
        "page" : "http://uictest.com/frames/",
        "referrer" : "http://uictest.com/",
        "screen" : {
            "devicePixelRatio" : 1,
            "deviceWidth" : 1920,
            "deviceHeight" : 1080,
            "deviceToolbarHeight" : 34,
```

```
            "width" : 942,
            "height" : 955,
            "orientation" : 0,
            "orientationMode" : "PORTRAIT"
        }
    }

}
```

This example shows the Client Environment message for Android and iOS:

```
"clientEnvironment": {
                "orientation": 90,
                "height": 720,
                "osVersion": "4.2.2",
                "pixelDensity": 2,
                "width": 1196,
                "deviceHeight": 360,
                "osType": "Android" or "iOS"
                "deviceWidth": 598
}
```

host
URL Host of the snapshot document.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.domCapture.host`

Description
URL Host of the snapshot document.

Value
URL host string.

Limitations
None

Dependencies
Used by Replay.

Example

This example shows a Type 12 DOM Capture message:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
```

```
                "url": "/h4/greeting.html"
            }
        ],
        "canvas": []
    }
    }
}
```

href
The destination of an anchor link.

Supported frameworks: IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.target.currState.href

Description
The destination of an anchor link.

Value
String value that indicates the destination (for example, http://www.example.com).

Limitations
None

Dependencies
None

Example

This example shows the target portion of a message:

```
"target": {
                "id": "[[\"HTML\",0],[\"BODY\",0],[\"A\",0]]",
                "idType": -2,
                "name": "",
                "tlType": "link",
                "type": "A",
                "subType": "",
                "currState": {
                    "innerText": "A link",
                    "href": "http://tealeaf.com"
                },
                "xPath": "[[\"HTML\",0],[\"BODY\",0],[\"A\",0]]",
                "isParentLink": false
            }
```

id
A unique identifier for the page instance. All hits from the page instance share an id value. A page reload or navigating away from the page resets the value.

Supported frameworks: IBM Tealeaf UI Capture

Path
<global>.<session>.id

Description
A unique identifier for the page instance. All hits from the page instance share an id value. A page reload or navigating away from the page resets the value.

Value
An integer corresponding to milliseconds since Jan 1, 1970 Coordinated Universal Time.

Limitations
None

Dependencies
Replay uses the id to identify all the events that belong to a single page instance.

Example

This example shows a portion of a message with the session id:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [...],
            "clientEnvironment": {...}
        }
    ]
}
```

id (target)

An opaque string that is used to uniquely identify the target HTML element. Uniqueness cannot be guaranteed by the library. It is the responsibility of the application developer to provide the elements with static and unique HTML id's or custom attributes. The Tealeaf library can then be configured to use either of these properties to uniquely identify the element. In the absence of these unique identifiers, the Tealeaf library defaults to using the XPath as the id. An XPath works in most situations but cannot guarantee to uniquely identify the element.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path

`<global>.<session>.<message>.target.id`

Description

An opaque string that is used to uniquely identify the target HTML element. Uniqueness cannot be guaranteed by the library. It is the responsibility of the application developer to provide the elements with static and unique HTML id's or custom attributes. The Tealeaf library can then be configured to use either of these properties to uniquely identify the element. In the absence of these unique identifiers, the Tealeaf library defaults to using the XPath as the id. An XPath works in most situations but cannot guarantee to uniquely identify the element.

Value

String

Limitations

None

Dependencies

Replay, Overstat, and Step-Based Eventing all use the id.

Example

This example shows a Type 4 Control message with an id used for the target:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {
```

```
                    "id": "bi",
                    "idType": -1,
                    "name": "buttonInput",
                    "type": "INPUT",
                    "subType": "button",
                    "position": {...},
                    "prevState": {...},
                    "currState": {...},
                    "isParentLink": false
                },
                "event": {
                    "type": "click"
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

idType
An enumeration that identifies the type of `id` being used by the library. The `idType` is only used by the browser-based UI Capture SDK.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.target.idType`

Description
An enumeration that identifies the type of `id` being used by the library. The `idType` is only used by the browser-based UI Capture SDK.

Value
Native id (for example, HTML 'id' attribute): -1, XPath: -2 for UIC, Custom attribute for UIC and Hashcode value for Native: -3, or XPath for Native iOS/Android: -4

Limitations
None

Dependencies
Replay uses this information with the target id to do a reverse lookup to get to the target element in the DOM.

Example

This example shows a Control (Type 4) message with an `id` that is derived from the XPath for Native iOS/Android, `idType` is (-4):

```
{
                    "screenviewOffset": 217,
                    "target": {
                        "id": "[KV,0]",
                        "position": {
                            "y": 368,
                            "x": 0,
                            "width": 360,
                            "height": 224
                        },
                        "idType": -4,
                        "currState": {
                            "y": "0",
                            "x": "0"
                        },
                        "style": {
                            "paddingTop": 3,
                            "paddingBottom": 0,
                            "paddingLeft": 0,
                            "hidden": false,
                            "paddingRight": 0
                        },
                        "subType": "View",
                        "type": "KeyboardView",
```

```
                        "tlType": "keyboard"
                    },
                    "type": 4,
                    "offset": 280,
                    "count": 1,
                    "fromWeb": false,
                    "event": {
                        "type": "UIKeyboardDidShowNotification",
                        "tlEvent": "kbDisplayed"
                    }
                },
```

This example shows a Control (Type 4) message with an `id` that is an assigned id, `idType` is (`-1`:

```
{
                    "count": 6,
                    "event": {
                        "tlEvent": "click",
                        "type": "Click"
                    },
                    "fromWeb": false,
                    "offset": 1664,
                    "screenviewOffset": 1601,
                    "target": {
                        "currState": {
                            "font": {
                                "bold": false,
                                "italic": false,
                                "name": "sans-serif",
                                "size": 36
                            },
                            "text": "Login"
                        },
                        "id": "com.ibm.tealeaf.aurora:id/login_button",
                        "idType": -1,
                        "position": {
                            "height": 96,
                            "width": 493,
                            "x": 113,
                            "y": 680
                        },
                        "style": {
                            "hidden": false,
                            "paddingBottom": 16,
                            "paddingLeft": 24,
                            "paddingRight": 24,
                            "paddingTop": 16,
                            "textAlign": "center",
                            "textAlphaColor": 255,
                            "textColor": 11711154
                        },
                        "subType": "TextView",
                        "tlType": "button",
                        "type": "Button"
                    },
                    "type": 4
                },
```

innerText
Value of display text in an anchor tag.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.target.currState.innerText`

Description
Value of display text in an anchor tag.

Value
String value that indicates the displayed text.

Limitations
None

Dependencies
None

Example

This example shows the target portion of a message:

```
"target": {
                "id": "[[\"HTML\",0],[\"BODY\",0],[\"A\",0]]",
                "idType": -2,
                "name": "",
                "tlType": "link",
                "type": "A",
                "subType": "",
                "currState": {
                    "innerText": "A link",
                    "href": "http://tealeaf.com"
                },
                "xPath": "[[\"HTML\",0],[\"BODY\",0],[\"A\",0]]",
                "isParentLink": false
            }
```

isParentLink
Used to indicate whether the target object has a parent button or <a> link.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.target.isParentLink`

Description
Value is set to `true` if the target object has a parent button or <a> link.

Value
Boolean

Limitations
None

Dependencies
None

Example

This example shows the target section of a message:

```
"target": {
                    "id": "bi",
                    "idType": -1,
                    "name": "buttonInput",
                    "type": "INPUT",
                    "subType": "button",
                    "position": {
                        "width": 67,
                        "height": 22,
                        "relXY": "0.5,0.3"
                    },
                    "prevState": {...},
                    "currState": {...},
                    "isParentLink": false
                },
                "event": {
                    "type": "click"
                }
```

libVersion
The library version number.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.clientEnvironment.webEnvironment.libVersion`

Description
The library version number.

Value
String of the format X.X.X.Y where X = 0-9 and Y is any positive integer (build number)

Limitations
None

Dependencies
None

Example

This example shows the sessions portion of a message with the library version:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [...],
            "clientEnvironment": {
                "webEnvironment": {
                    "libVersion": "1.0.0.152",
                    "page": "http://moksha.tealeaf.com/html4.html#",
                    "windowId": "I7Gf$t",
                    "screen": {
                        "orientation": 0,
                        "orientationMode": "PORTRAIT"
                    }
                }
            }
        }
    ]
}
```

line
The line number where the exception occurred.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.exception.line`

Description
The line number where the exception occurred.

Value
Number

Limitations
None

Dependencies
None.

Example

This example shows a Type 6 Exception message:

```
{
                    "type" : 6,
                    "offset" : 4606,
                    "screenviewOffset" : 4603,
                    "count" : 3,
                    "fromWeb" : true,
                    "exception" : {
                        "description" : "Uncaught TypeError: Cannot read property
'type' of undefined",
                        "url" : "http://www.xyz.com/js/badscript.js",
                        "line" : 258
                    }
}
```

messageVersion
The JSON Format version number.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.messageVersion`

Description
The JSON Format version number.

Value
String of the format X.X.X.X where X = 0-9.

Limitations
None

Dependencies
None

Example

This example shows the message version:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [...],
            "clientEnvironment": {...}
        }
    ]
}
```

name
The value of the name attribute of the target HTML element.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.target.name`

Description
The value of the name attribute of the target HTML element.

Value
String

Limitations
The underlying HTML element must have a name attribute.

Dependencies
Replay and Step-Based eventing use this property.

Example

This example shows a Type 4 Control message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {
                    "id": "bi",
                    "idType": -1,
                    "name": "buttonInput",
                    "type": "INPUT",
                    "subType": "button",
                    "position": {...},
                    "prevState": {...},
                    "currState": {...},
                    "isParentLink": false
                },
                "event": {
                    "type": "click"
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

offset
The message's time that is offset in milliseconds since the start of the session (`<global>.<session>.startTime`)

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.offset`

Description
The message's time that is offset in milliseconds since the start of the session (`<global>.<session>.startTime`)

Value
Integer.

Limitations
None

Dependencies
None

Example

This example shows a Type 4 Control message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {...},
                "event": {...}
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

orientation
This is the original orientation of the screen.,

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.orientation`

Description
The original orientation of the screen.

Value
An integer. Valid values are 0, 90, 180, or -90.

Limitations
An integer. Valid values are 0, 90, 180, or -90.

Dependencies
Used by templates to generate the correct html page to dislay in BBR.

Example

This example shows the Client Environment message for UIC:

```
clientEnvironment" : {
    "webEnvironment" : {
        "libVersion" : "5.0.0.XXXX",
        "page" : "http://uictest.com/frames/",
        "referrer" : "http://uictest.com/",
        "screen" : {
            "devicePixelRatio" : 1,
            "deviceWidth" : 1920,
            "deviceHeight" : 1080,
            "deviceToolbarHeight" : 34,
            "width" : 942,
            "height" : 955,
            "orientation" : 0,
            "orientationMode" : "PORTRAIT"
        }
    }

}
```

This example shows the Client Environment message for Android and iOS:

```
"clientEnvironment": {
                "orientation": 90,
                "height": 720,
                "osVersion": "4.2.2",
                "pixelDensity": 2,
                "width": 1196,
                "deviceHeight": 360,
                "osType": "Android" or "iOS"
                "deviceWidth": 598
}
```

pageHeight
The HTML document height in CSS pixels.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.clientState.pageHeight`

Description
The HTML document height in CSS pixels.

Value
Integer.

Limitations
None.

Dependencies
None

Example

This example shows a Type 1 Client State message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 1,
                "offset": 7353,
                "screenviewOffset": 7352,
                "count": 2,
                "fromWeb": true,
                "clientState": {
                    "pageWidth": 954,
                    "pageHeight": 820,
                    "viewPortWidth": 877,
                    "viewPortHeight": 836,
                    "viewPortX": 0,
                    "viewPortY": 0,
                    "event": "load",
                    "viewTime": 0
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

pageWidth
The HTML document width in CSS pixels.

Supported frameworks: IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.clientState.pageWidth

Description
The HTML document width in CSS pixels.

Value
Integer.

Limitations
None.

Dependencies
None

Example

This example shows a Type 1 Client State message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 1,
                "offset": 7353,
                "screenviewOffset": 7352,
                "count": 2,
                "fromWeb": true,
                "clientState": {
                    "pageWidth": 954,
                    "pageHeight": 820,
                    "viewPortWidth": 877,
                    "viewPortHeight": 836,
                    "viewPortX": 0,
                    "viewPortY": 0,
                    "event": "load",
                    "viewTime": 0
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

position
An object that contains information about the location of the pointing device and the target object's width and height within the context of a user action message.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.target.position

Description
An object that contains information about the location of the pointing device and the target object's width and height within the context of a user action message.

Value
Object containing width, height, and relXY properties. Refer to individual property descriptions for details.

Limitations
None

Dependencies
cxOverstat uses this information for producing heat maps.

Example

This example shows a Type 4 Control message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {
                    "id": "bi",
                    "idType": -1,
                    "name": "buttonInput",
                    "type": "INPUT",
                    "subType": "button",
                    "position": {
                        "width": 67,
                        "height": 22,
                        "relXY": "0.5,0.3"
                    },
                    "prevState": {...},
                    "currState": {...},
                    "isParentLink": false
                },
                "event": {
                    "type": "click"
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

referrer (Screenview)
The name of the previous Screenview, if any, of the page that generated the JSON message. If there is no referrer, this field is a blank string. This field is in iOS and Android Screenview messages, not UIC.

Path
`<message>.<screenview>.referrer`

Description
The name of the previous Screenview, if any, of the page that generated the JSON message. If there is no referrer, this field is a blank string.

Value
String.

Limitations
None

Dependencies
Used by Replay.

Example

This example shows a Screenview message:

```
{
    "offset": 124,
    "contextOffset": 4556,
    "type": 2,
    "context": {
            "type": "LOAD",
            "name": "PAGE 2",
            "referrer": "PAGE 1"
        }
}

{
    "type": 2,
    "offset": 19216

    "context": {
        "type": "UNLOAD",
        "name": "PAGE 2"
    }
}

{
    "type": 2,
    "offset": 2144,
    "contextOffset": 0,
    "count": 9,
    "fromWeb": true,

    "webviewId": "webview1",
    "screenview": {
        "type": "LOAD",
        "name": "Ford",
        "url": "/dynamic/ford.aspx",

        "host": "http://www.cartest.com",
        "referrer": "BMW",
        "referrerUrl": "/dynamic/bmw.aspx"
    }
}
```

referrer (webEnvironment)
The HTTP referrer, if any, of the page that generated the JSON message. If there is no referrer this is a blank string.

Supported frameworks: IBM Tealeaf UI Capture

Path
`clientEnvironment.webEnvironment.referrer`

Description
The HTTP referrer, if any, of the page. If there is no referrer this is a blank string.

Value
String.

Limitations
None

Dependencies
Used by Replay.

Example

This example shows the UIC Client Environment message:

```
"clientEnvironment" : {
    "webEnvironment" : {
        "libVersion" : "5.0.0.XXXX",
        "page" : "http://uictest.com/frames/",
        "referrer" : "http://uictest.com/",
        "screen" : {
```

```
            "devicePixelRatio" : 1,
            "deviceWidth" : 1920,
            "deviceHeight" : 1080,
            "deviceToolbarHeight" : 34,
            "width" : 942,
            "height" : 955,
            "orientation" : 0,
            "orientationMode" : "PORTRAIT"
        }

    }

}
```

relXY
The relative X and Y co-ordinate of the pointing device within the target object that is calculated as a fraction of the width and height of the target. The fractional value is rounded off to a single decimal. This property is generated only if a click event is recorded.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.target.position.relXY`

Description
The relative X and Y co-ordinate of the pointing device within the target object that is calculated as a fraction of the width and height of the target. The fractional value is rounded off to a single decimal.

Value
String of the format "X,Y" where X = 0.0 - 1.0 and Y = 0.0 - 1.0. Default value is `0.5,0.5`.

Limitations
None

Dependencies
cxOverstat uses this information for producing heat maps.
This property is generated only if a click event is recorded.

Example

This example shows a Type 4 Control message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {
                    "id": "bi",
                    "idType": -1,
                    "name": "buttonInput",
                    "type": "INPUT",
                    "subType": "button",
                    "position": {
                        "width": 67,
                        "height": 22,
                        "relXY": "0.5,0.3"
                    },
                    "prevState": {...},
                    "currState": {...},
                    "isParentLink": false
                },
                "event": {
                    "type": "click"
```

```
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

root
Serialized HTML of the document.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.domCapture.root`

Description
Serialized HTML of the document.

Value

HTML

Limitations
None

Dependencies

Used by Tealeaf Eventing and Replay.

Example

This example shows a Type 12 DOM Capture message:

```
{
    // DOM Capture messages use type 12
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
                "url": "/h4/greeting.html"
            }
        ],
        "canvas": []
    }
    }
}
```

screenviewOffset
The message's time that is offset in milliseconds since the start of the screenview with which this message is associated.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.screenviewOffset`

Description
The message's time that is offset in milliseconds since the start of the screenview with which this message is associated.

Value
Integer.

Limitations
None

Dependencies
None

Example

This example shows a Type 4 Control message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {...},
                "event": {...}
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

scrollX
The number of pixels that the document is currently scrolled from the left.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.gesture.scrollx`

Description
The number of pixels that the document is currently scrolled from the left.

Value
Integer

Limitations
None

Dependencies
Used by Replay.

Example

This example shows a Type 11 Gesture message:

```
{
  "fromWeb": false,
  "type": 11,
  "offset": 46788,
  "screenviewOffset": 42208,
  "count": 14,
```

```
  "event": {
    "type": "ACTION_DOWN",
    "tlEvent": "tap"
  },
  "touches": [
    [
      {
        "position": {
          "x": 179,
          "y": 543
        },
        "control": {
          "position": {
            "height": 184,
            "width": 1080,
            "relXY": "0.17,0.93"
              "scrollX": 10
              "scrollY": 15
          },
          "id": "[RL,0]",
          "idType": -4,
          "type": "RelativeLayout",
          "subType": "ViewGroup",
          "tlType": "canvas"
        }
      }
    ]
  ]
}
```

scrollY
The number of pixels that the document is currently scrolled from the top.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.gesture.scrolly`

Description
The number of pixels that the document is currently scrolled from the top.

Value
Integer

Limitations
None

Dependencies
Used by Replay.

Example

This example shows a Type 11 Gesture message:

```
{
  "fromWeb": false,
  "type": 11,
  "offset": 46788,
  "screenviewOffset": 42208,
  "count": 14,
  "event": {
    "type": "ACTION_DOWN",
    "tlEvent": "tap"
  },
  "touches": [
    [
      {
        "position": {
          "x": 179,
          "y": 543
        },
        "control": {
          "position": {
            "height": 184,
            "width": 1080,
            "relXY": "0.17,0.93"
```

```
                "scrollX": 10
                "scrollY": 15
            },
            "id": "[RL,0]",
            "idType": -4,
            "type": "RelativeLayout",
            "subType": "ViewGroup",
            "tlType": "canvas"
          }
        }
      ]
    ]
}
```

serialNumber
The serial number of the payload from a specific page instance. The first hit from a new page instance always begins with a serial number of 1. Every subsequent hit from the same page instance increments the serial number by 1. A page reload or navigating away from the page resets the value.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.serialNumber`

Description
The serial number of the payload from a specificgiven page instance. The first hit from a new page instance always begins with a serial number of 1. Every subsequent hit from the same page instance increments the serial number by 1. A page reload or navigating away from the page resets the value.

Value
A positive integer.

Limitations
None

Dependencies
Replay uses this property to order and combine the events in the correct sequence.

Example

This example shows the sessions portion of a message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [...],
            "clientEnvironment": {...}
        }
    ]
}
```

startTime
An absolute time stamp (milliseconds since Jan 1, 1970 Coordinated Universal Time) indicating when the current page instance (session) was started. A page reload or navigating away from the page resets the value.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.startTime`

Description
An absolute time stamp (milliseconds since Jan 1, 1970 Coordinated Universal Time) indicating when the current page instance (session) was started. A page reload or navigating away from the page resets the value.

Value
An integer.

Limitations
None

Dependencies
None

Example

This example shows the sessions portion of a message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [...],
            "clientEnvironment": {...}
        }
    ]
}
```

subType
String containing the HTML type of input element.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.target.subType`

Description
String containing the HTML type of input element.

Value
String containing the valid HTML INPUT element types such as **check box**, **radio**, **text**, etc.

Reference
http://www.w3.org/TR/html401/interact/forms.html#h-17.4.1

Limitations
Valid only for HTML INPUT element type.

Dependencies
None

Example

This example shows a Type 4 Control message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
```

```
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {
                    "id": "bi",
                    "idType": -1,
                    "name": "buttonInput",
                    "type": "INPUT",
                    "subType": "button",
                    "position": {...},
                    "prevState": {...},
                    "currState": {...},
                    "isParentLink": false
                },
                "event": {
                    "type": "click"
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

target
An object that contains information about the HTML element that is the target of a user action.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.target`

Description
An object that contains information about the HTML element that is the target of a user action.

Value
Object. Refer to individual property descriptions for details.

Limitations
The target object is present only in control message types (`<global>.<session>.<message>.type = 4`)

Dependencies
Replay, Overstat, and Step-Based Eventing use various members of the target object.

Example

This example shows a Type 4 Control message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {
                    "id": "bi",
                    "idType": -1,
                    "name": "buttonInput",
                    "type": "INPUT",
                    "subType": "button",
                    "position": {
                        "width": 67,
                        "height": 22,
```

```
                        "relXY": "0.5,0.3"
                    },
                    "prevState": {...},
                    "currState": {...},
                    "isParentLink": false
                },
                "event": {
                    "type": "click"
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

timezoneOffset
The difference, in minutes, between UTC and local time. This means that the offset is positive if the local timezone is behind UTC and negative if it is ahead. Changing the timezone settings on the machine might cause the timezoneOffset to change within the session. Daylight Saving Time prevents this value from being a constant even for a specific locale.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.timezoneOffset`

Description
The difference, in minutes, between UTC and local time. This means that the offset is positive if the local timezone is behind UTC and negative if it is ahead. Changing the timezone settings on the machine might cause the timezoneOffset to change within the session. Daylight Saving Time prevents this value from being a constant even for a specific locale.

Value
An integer value corresponding to the difference, in minutes, between UTC (timezoneOffset = 0) and local time.

Limitations
None

Dependencies
None

Example

This example shows the sessions portion of a message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [...],
            "clientEnvironment": {...}
        }
    ]
}
```

type (message)
Enumeration that is used to identify the type of message. The message type determines the subsequent properties that would be populated within the message object.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.type`

Description
Enumeration that is used to identify the type of message. The message type determines the subsequent properties that would be populated within the message object.

Value
Integer.

- 1 - Client State
- 2 - Screenview
- 3 - Connection (not used by UIC)
- 4 - Control
- 5 - Custom
- 6 - Exception
- 7 - Performance
- 8 - Web storage
- 9 - Overstat hover event
- 10 - Layout
- 11 - Gesture
- 12 - DOM Capture
- 13 - Geolocation

Limitations
None

Dependencies
Replay, Overstat, Step-Based Eventing use this value.

Example

This example shows a Type 4 Control message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {...},
                "event": {...}
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

type (event)
An enumeration that contains information about the nature of the user action.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.event.type`

Description
An enumeration that contains information about the nature of the user action.

Value
String.

- click - When a user clicks or taps on a non-input type HTML element such as a link, div, image. Click actions on input type HTML elements are consolidated into change or blur messages.
- change - A user action with an input type HTML element results in a state change. For example, when text is entered into a textbox or a checkbox is toggled.
- blur - A user action with an input type HTML element does not result in a state change. For example, when the user tabs through a text box without changing it's content etc.
- orientationchange - A user action that changes the orientation on browsers and platforms supporting portrait and landscape modes.
- touchend - A user action that provides pinch gesture information on browsers and platforms that have gesture support.

Limitations
None.

Dependencies
Replay and Overstat use this property to determine the nature of user interaction.

Example

This example shows a Type 4 Control message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {...},
                "event": {
                    "type": "click"
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

type (target)
The HTML element tag of the target element.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.target.type`

Description
The HTML element tag of the target element.

Value
String containing any of the valid HTML element types such as "A", "INPUT", "DIV" etc.

Limitations
None

Dependencies
None

Example

This example shows a Type 4 Control message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {
                    "id": "bi",
                    "idType": -1,
                    "name": "buttonInput",
                    "type": "INPUT",
                    "subType": "button",
                    "position": {...},
                    "prevState": {...},
                    "currState": {...},
                    "isParentLink": false
                },
                "event": {
                    "type": "click"
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

url
URL path of the snapshot document.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.domCapture.url`

Description
URL path of the snapshot document.

Value
URL path string.

Limitations
None

Dependencies
Used by Replay.

Example

This example shows a Type 12 DOM Capture message:

```
{
    // DOM Capture messages use type 12
```

```
    "type": 12,

    // The standard UIC message properties
    "offset": 16821,
    "screenviewOffset": 16817,
    "count": 5,
    "fromWeb": true,

    "domCapture": {
        "dcid": "dcid-3"
        "charset": "ISO-8859-1",
        "root": "<html><body><iframe id="greeting.html" tltid="tlt-4"/>
</body></html>",
        "host": "http://www.uictest.com",
        "url": "/h4/dcTest.html",
        "eventOn": true,
        "frames": [
            {
                "tltid": "tlt-4",
                "root": "<html><body>Hello, World!</body></html>",
                "charset": "ISO-8859-1",
                "host": "http://www.uictest.com",
                "url": "/h4/greeting.html"
            }
        ],
        "canvas": []
    }
    }
}
```

viewPortHeight
The viewport height in CSS pixels.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.clientState.viewPortHeight`

Description
The viewport height in CSS pixels.

Value
Integer.

Limitations
None.

Dependencies
None

Example

This example shows a Type 1 Client State message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 1,
                "offset": 7353,
                "screenviewOffset": 7352,
                "count": 2,
                "fromWeb": true,
                "clientState": {
                    "pageWidth": 954,
                    "pageHeight": 820,
                    "viewPortWidth": 877,
                    "viewPortHeight": 836,
                    "viewPortX": 0,
```

```
                    "viewPortY": 0,
                    "event": "load",
                    "viewTime": 0
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

viewPortWidth
The viewport width in CSS pixels.

Supported frameworks: IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.clientState.viewPortWidth

Description
The viewport width in CSS pixels.

Value
Integer.

Limitations
None.

Dependencies
None

Example

This example shows a Type 1 Client State message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 1,
                "offset": 7353,
                "screenviewOffset": 7352,
                "count": 2,
                "fromWeb": true,
                "clientState": {
                    "pageWidth": 954,
                    "pageHeight": 820,
                    "viewPortWidth": 877,
                    "viewPortHeight": 836,
                    "viewPortX": 0,
                    "viewPortY": 0,
                    "event": "load",
                    "viewTime": 0
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

viewPortX
The viewport location X coordinate.

Supported frameworks: IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.clientState.viewPortX

Description
The viewport location X coordinate of the upper left corner.

Value
Integer.

Limitations
None.

Dependencies
None

Example

This example shows a Type 1 Client State message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 1,
                "offset": 7353,
                "screenviewOffset": 7352,
                "count": 2,
                "fromWeb": true,
                "clientState": {
                    "pageWidth": 954,
                    "pageHeight": 820,
                    "viewPortWidth": 877,
                    "viewPortHeight": 836,
                    "viewPortX": 0,
                    "viewPortY": 0,
                    "event": "load",
                    "viewTime": 0
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

viewPortY
The viewport location Y coordinate.

Supported frameworks: IBM Tealeaf UI Capture

Path
<global>.<session>.<message>.clientState.viewPortY

Description
The viewport location Y coordinate of the upper left corner.

Value
Integer.

Limitations
None.

Dependencies
None

Example

This example shows a Type 1 Client State message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 1,
                "offset": 7353,
                "screenviewOffset": 7352,
                "count": 2,
                "fromWeb": true,
                "clientState": {
                    "pageWidth": 954,
                    "pageHeight": 820,
                    "viewPortWidth": 877,
                    "viewPortHeight": 836,
                    "viewPortX": 0,
                    "viewPortY": 0,
                    "event": "load",
                    "viewTime": 0
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

viewTime
The view time in milliseconds.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.clientState.viewTime`

Description
The viewport location X coordinate.

Value
Integer.

Limitations
None.

Dependencies
None

Example

This example shows a Type 1 Client State message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 1,
                "offset": 7353,
                "screenviewOffset": 7352,
```

```
                "count": 2,
                "fromWeb": true,
                "clientState": {
                    "pageWidth": 954,
                    "pageHeight": 820,
                    "viewPortWidth": 877,
                    "viewPortHeight": 836,
                    "viewPortX": 0,
                    "viewPortY": 0,
                    "event": "load",
                    "viewTime": 0
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

webviewId
Used to identify which webview the message came from. This field is only used when `fromWeb` is true and the application is a hybrid application.

Supported frameworks: IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.webviewId`

Description
Used to identify which webview the message came from. This field can appear in any JSON message where `fromWeb` is true and the application is a hybrid application.

Value
webview*n* where *n* is the identifier for the webview that generated the JSON.

Limitations
None

Dependencies
BBR uses this to determine which webview produced the JSON and to know what to highlight or select.

Example

This example shows an Application Context (Type 2) message with the `webivewId` field:

```
 {
                    "count": 2,
                    "dcid": "dcid-1.1429298094152s",
                    "fromWeb": true,
                    "offset": 2010,
                    "screenview": {
                        "host": "file://",
                        "name": "root",
                        "referrer": "",
                        "type": "LOAD",
                        "url": "/android_asset/mobile_domcap/
embeddedGesturesMenu.html"
                    },
                    "screenviewOffset": 0,
                    "type": 2,
                    "webviewId": "tl.hybridhtmlembedded:id/myWebView1"
                },
```

width
The width in CSS pixels of the target object.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.target.position.width`

Description
The width in CSS pixels of the target object.

Value
Integer

Limitations
None

Dependencies
Overstat uses this information for producing heat maps.

Example

This example shows a type 4 control message:

```
{
    "messageVersion": "1.1.0.0",
    "serialNumber": 2,
    "sessions": [
        {
            "id": "ID13H56M10S772R0.955239302458247",
            "startTime": 1343076970772,
            "timezoneOffset": 420,
            "messages": [
            {
                "type": 4,
                "offset": 17329,
                "screenviewOffset": 17327,
                "count": 8,
                "fromWeb": true,
                "target": {
                    "id": "bi",
                    "idType": -1,
                    "name": "buttonInput",
                    "type": "INPUT",
                    "subType": "button",
                    "position": {
                        "width": 67,
                        "height": 22,
                        "relXY": "0.5,0.3"
                    },
                    "prevState": {...},
                    "currState": {...},
                    "isParentLink": false
                },
                "event": {
                    "type": "click"
                }
            }],
            "clientEnvironment": {...}
        }
    ]
}
```

width (Client Environment)
The Initial session width of the display/viewport divided by pixel density.

Supported frameworks: Android, iOS, IBM Tealeaf UI Capture

Path
`<global>.<session>.<message>.clientState.width`

Description
The initial session width of the display/viewport divided by pixel density.

Value
An integer.

Limitations
None

Dependencies
Used by replay

Example

This example shows the Client Environment message for UIC:

```
clientEnvironment" : {
    "webEnvironment" : {
        "libVersion" : "5.0.0.XXXX",
        "page" : "http://uictest.com/frames/",
        "referrer" : "http://uictest.com/",
        "screen" : {
            "devicePixelRatio" : 1,
            "deviceWidth" : 1920,
            "deviceHeight" : 1080,
            "deviceToolbarHeight" : 34,
            "width" : 942,
            "height" : 955,
            "orientation" : 0,
            "orientationMode" : "PORTRAIT"
        }
    }

}
```

This example shows the Client Environment message for Android and iOS:

```
"clientEnvironment": {
                "orientation": 90,
                "height": 720,
                "osVersion": "4.2.2",
                "pixelDensity": 2,
                "width": 1196,
                "deviceHeight": 360,
                "osType": "Android" or "iOS"
                "deviceWidth": 598
}
```

Default client framework objects
Each client framework has a default set of JSON objects. The objects map to System Step Attributes provided by Tealeaf.

Data sources for JSON objects

This table maps JSON attributes submitted by one or more client frameworks to the System Step Attributes provided by Tealeaf. These attributes are available in the `System Step Attributes` label in the Hit Attributes tab of the Event Manager.

Table 16. Default Client Framework Objects

Step Attribute Path	UIC(j2)	IBM Tealeaf cxOverstat (only)	iOS	Android
`.messageVersion`	Y		Y	Y
`.serialNumber`	Y		Y	Y
`.sessions[0].clientEnvironment.osVersion`			Y	Y

Table 16. Default Client Framework Objects (continued)

Step Attribute Path	UIC(j2)	IBM Tealeaf cxOverstat (only)	iOS	Android
`.sessions[0].clientEnvironment.height`			Y	Y
`.sessions[0].clientEnvironment.width`			Y	Y
`.sessions[0].clientEnvironment.mobileEnvironment.appName`			Y	Y
`.sessions[0].clientEnvironment.mobileEnvironment.totalMemory`			Y	Y
`.sessions[0].clientEnvironment.mobileEnvironment.totalStorage`			Y	Y
`.sessions[0].clientEnvironment.mobileEnvironment.orientationType`			Y	Y
`.sessions[0].clientEnvironment.mobileEnvironment.appVersion`			Y	Y
`.sessions[0].clientEnvironment.mobileEnvironment.manufacturer`			Y	Y
`.sessions[0].clientEnvironment.mobileEnvironment.userId`			Y	Y
`.sessions[0].clientEnvironment.mobileEnvironment.locale`			Y	Y
`.sessions[0].clientEnvironment.mobileEnvironment.deviceModel`			Y	Y
`.sessions[0].clientEnvironment.mobileEnvironment.language`			Y	Y
`.sessions[0].clientEnvironment.mobileEnvironment.android.brand`				Y
`.sessions[0].clientEnvironment.mobileEnvironment.android.fingerPrint`				Y
`.sessions[0].clientEnvironment.mobileEnvironment.android.keyboardType`				Y

Table 16. Default Client Framework Objects (continued)

Step Attribute Path	UIC(j2)	IBM Tealeaf cxOverstat (only)	iOS	Android
`.sessions[0].clientEnvironment.webEnvironment.libVersion`	Y			
`.sessions[0].clientEnvironment.webEnvironment.page`	Y			
`.sessions[0].clientEnvironment.webEnvironment.screen.orientation`	Y			
`.sessions[0].clientEnvironment.webEnvironment.screen.orientationMode`	Y			
`.sessions[0].id`	Y			
`.sessions[0].message.clientState.event`	Y			
`.sessions[0].message.clientState.pageHeight`	Y	Y		
`.sessions[0].message.clientState.pageWidth`	Y	Y		
`.sessions[0].message.clientState.viewPortHeight`	Y			
`.sessions[0].message.clientState.viewPortWidth`	Y			
`.sessions[0].message.clientState.viewPortX`	Y			
`.sessions[0].message.clientState.viewPortXStart`	Y			
`.sessions[0].message.clientState.viewPortY`	Y			
`.sessions[0].message.clientState.viewPortYStart`	Y			
`.sessions[0].message.clientState.viewTime`	Y			
`.sessions[0].message.connection.description`			Y	Y

Table 16. Default Client Framework Objects (continued)

Step Attribute Path	UIC(j2)	IBM Tealeaf cxOverstat (only)	iOS	Android
`.sessions[0].message.connection.initTime`			Y	Y
`.sessions[0].message.connection.loadTime`			Y	Y
`.sessions[0].message.connection.responseDataSize`			Y	Y
`.sessions[0].message.connection.responseTime`			Y	Y
`.sessions[0].message.connection.statusCode`			Y	Y
`.sessions[0].message.connection.url`			Y	Y
`.sessions[0].message.count`	Y			
`.sessions[0].message.customEvent.name`	Y		Y	Y
`.sessions[0].message.customEvent.data.[property based on custom event]`	Y		Y	Y
`.sessions[0].message.event.tlEvent`	Y		Y	Y
`.sessions[0].message.event.type`	Y		Y	Y
`.sessions[0].message.event.subType`	Y		Y	Y
`.sessions[0].message.exception.description`	Y		Y	Y
`.sessions[0].message.exception.line`	Y			
`.sessions[0].message.exception.name`			Y	Y
`.sessions[0].message.exception.stackTrace`			Y	Y

Table 16. Default Client Framework Objects (continued)

Step Attribute Path	UIC(j2)	IBM Tealeaf cxOverstat (only)	iOS	Android
`.sessions[0].message.exception.url`	Y			
`.sessions[0].message.focusInOffset`	Y		Y	Y
`.sessions[0].message.fromWeb`	Y		Y	Y
`.sessions[0].message.mobileState.orientation`			Y	Y
`.sessions[0].message.mobileState.freeStorage`			Y	Y
`.sessions[0].message.mobileState.battery`			Y	Y
`.sessions[0].message.mobileState.freeMemory`			Y	Y
`.sessions[0].message.mobileState.connectionType`			Y	Y
`.sessions[0].message.mobileState.carrier`			Y	Y
`.sessions[0].message.mobileState.networkReachability`			Y	Y
`.sessions[0].message.mobileState.ip`			Y	Y
`.sessions[0].message.mobileState.androidState.keyboardState`				Y
`.sessions[0].message.offset`	Y		Y	Y
`.sessions[0].message.performance.navigation.redirectCount`	Y			
`.sessions[0].message.performance.navigation.type`	Y			

Table 16. Default Client Framework Objects (continued)

Step Attribute Path	UIC(j2)	IBM Tealeaf cxOverstat (only)	iOS	Android
`.sessions[0].message. performance.timing. connectEnd`	Y			
`.sessions[0].message. performance.timing. connectStart`	Y			
`.sessions[0].message. performance.timing. domainLookupEnd`	Y			
`.sessions[0].message. performance. timing.domainLookupStart`	Y			
`.sessions[0].message. performance.timing. domComplete`	Y			
`.sessions[0].message. performance.timing. domContentLoadedEventEnd`	Y			
`.sessions[0].message. performance.timing. domContentLoadedEventStart`	Y			
`.sessions[0].message. performance.timing. domInteractive`	Y			
`.sessions[0].message. performance.timing. domLoading`	Y			
`.sessions[0].message. performance.timing. fetchStart`	Y			
`.sessions[0].message. performance.timing. loadEventEnd`	Y			
`.sessions[0].message. performance.timing. loadEventStart`	Y			
`.sessions[0].message. performance.timing. navigationStart`	Y			

Table 16. Default Client Framework Objects (continued)

Step Attribute Path	UIC(j2)	IBM Tealeaf cxOverstat (only)	iOS	Android
`.sessions[0].message.performance.timing.redirectEnd`	Y			
`.sessions[0].message.performance.timing.redirectStart`	Y			
`.sessions[0].message.performance.timing.responseEnd`	Y			
`.sessions[0].message.performance.timing.responseStart`	Y			
`.sessions[0].message.performance.timing.secureConnectionStart`	Y			
`.sessions[0].message.performance.timing.unloadEventEnd`	Y			
`.sessions[0].message.performance.timing.unloadEventStart`	Y			
`.sessions[0].message.screenview.name`	Y	Y	Y	Y
`.sessions[0].message.screenview.referrer`	Y	Y	Y	Y
`.sessions[0].message.screenview.renderTime`		Y		
`.sessions[0].message.screenview.type`	Y	Y	Y	Y
`.sessions[0].message.screenview.url`	Y	Y		
`.sessions[0].message.screenviewOffset`	Y	Y	Y	Y
`.sessions[0].message.target.currState.[property based on control]`	Y		Y	Y

Table 16. Default Client Framework Objects (continued)

Step Attribute Path	UIC(j2)	IBM Tealeaf cxOverstat (only)	iOS	Android
`.sessions[0].message.target.dwell`	Y		Y	Y
`.sessions[0].message.target.id`	Y	Y	Y	Y
`.sessions[0].message.target.idType`	Y	Y		
`.sessions[0].message.target.isParentLink`	Y	Y		
`.sessions[0].message.target.name`	Y			
`.sessions[0].message.target.position.height`	Y		Y	Y
`.sessions[0].message.target.position.relXY`	Y	Y		
`.sessions[0].message.target.position.width`	Y		Y	Y
`.sessions[0].message.target.position.x`			Y	Y
`.sessions[0].message.target.position.y`			Y	Y
`.sessions[0].message.target.prevState.[property based on control]`	Y		Y	Y
`.sessions[0].message.target.subType`	Y		Y	Y
`.sessions[0].message.target.tlType`	Y		Y	Y
`.sessions[0].message.target.type`	Y		Y	Y
`.sessions[0].message.target.visitedCount`	Y		Y	Y
`.sessions[0].message.type`	Y		Y	Y

Table 16. Default Client Framework Objects (continued)				
Step Attribute Path	**UIC(j2)**	**IBM Tealeaf cxOverstat (only)**	**iOS**	**Android**
`.sessions[0].startTime`	Y		Y	Y
`.sessions[0].timezoneOffset`	Y			

Mappings between JSON properties to Tealeaf System Objects

In the table below, you can review how the JSON properties are mapped to hit attributes, events, and dimensions that are provided by Tealeaf.

Table 17. Default Client Framework Objects			
Step Attribute Path	**System Step Attribute**	**As condition for Event**	**feeds Dimension(s)**
`.messageVersion`	"Step - SDK Version" in the *IBM Tealeaf Event Manager Manual*		
`.serialNumber`	"Step - Serial Number" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].clientEnvironment.webEnvironment.libVersion`	"Step - Client Environment Library Version" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].clientEnvironment.webEnvironment.page`	"Step - Client Environment Page" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].clientEnvironment.webEnvironment.screen.orientation`	"Step - Client Environment Screen Orientation" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].clientEnvironment.webEnvironment.screen.orientationMode`	"Step - Client Environment Screen Orientation Mode" in the *IBM Tealeaf Event Manager Manual*		

Table 17. Default Client Framework Objects (continued)

Step Attribute Path	System Step Attribute	As condition for Event	feeds Dimension(s)
`.sessions[0].message.clientState.event`	"Step - ClientState Event" in the *IBM Tealeaf Event Manager Manual*	"Step - Usability Attention Map Viewport Height" in the *IBM Tealeaf Event Manager Manual*; "Step - Usability Attention Map Y View Time" in the *IBM Tealeaf Event Manager Manual*; "Step - Usability Focal Slice Y (BB)" in the *IBM Tealeaf Event Manager Manual*	"Step - Usability Focal Slice Y" in the *IBM Tealeaf Event Manager Manual*
`.sessions[0].message.clientState.pageHeight`	"Step - ClientState Page Height" in the *IBM Tealeaf Event Manager Manual*	"Step - Usability Attention Map Viewport Height" in the *IBM Tealeaf Event Manager Manual*; "Step - Usability Focal Slice Y (BB)" in the *IBM Tealeaf Event Manager Manual*	"Step - Usability Focal Slice Y" in the *IBM Tealeaf Event Manager Manual*
`.sessions[0].message.clientState.pageWidth`	"Step - ClientState Page Width" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.clientState.viewPortHeight`	"Step - ClientState Viewport Height" in the *IBM Tealeaf Event Manager Manual*	"Step - Usability Attention Map Viewport Height (BB)" in the *IBM Tealeaf Event Manager Manual*; "Step - Usability Attention Map Y View Time" in the *IBM Tealeaf Event Manager Manual*; "Step - Usability Focal Slice Y (BB)" in the *IBM Tealeaf Event Manager Manual*	"Step - Usability Focal Slice Y" in the *IBM Tealeaf Event Manager Manual*; "Step - Usability View Port Height" in the *IBM Tealeaf Event Manager Manual*
`.sessions[0].message.clientState.viewPortWidth`	"Step - ClientState Viewport Width" in the *IBM Tealeaf Event Manager Manual*		

Table 17. Default Client Framework Objects (continued)

Step Attribute Path	System Step Attribute	As condition for Event	feeds Dimension(s)
`.sessions[0].message.clientState.viewPortX`	"Step - ClientState Viewport X" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.clientState.viewPortY`	"Step - ClientState Viewport Y" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.clientState.viewTime`	"Step - ClientState View Time" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.connection.description`	"Step - Connection Description" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.connection.initTime`	"Step - Connection Init Time" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.connection.loadTime`	"Step - Connection Load Time" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.connection.responseDataSize`	"Step - Connection Response Data Size" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.connection.responseTime`	"Step - Connection Response Time" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.connection.statusCode`	"Step - Connection Status Code" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.connection.url`	"Step - Connection URL" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.screenViewOffset`			

Table 17. Default Client Framework Objects (continued)

Step Attribute Path	System Step Attribute	As condition for Event	feeds Dimension(s)
`.sessions[0].message.count`	"Step - Message Count" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.event.tlEvent`			
`.sessions[0].message.event.type`	"Step - TL Event Type" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.event.subType`	"Step - Event Type" in the *IBM Tealeaf Event Manager Manual*	"Step - Usability Form Field Visit" in the *IBM Tealeaf Event Manager Manual*	
`.sessions[0].message.focusInOffset`	"Step - Focus Offset" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.fromWeb`	"Step - Message From Web" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.objects.performance.loadEventStart`	"Step - Performance URL Render Time" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.objects.performance.redirectCount`	"Step - Objects Performance Redirect Count" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.objects.performance.referrer`	"Step - Objects Performance Referrer" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.offset`	"Step - Offset" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.performance.navigation.type`	"Step - Performance Navigation Type" in the *IBM Tealeaf Event Manager Manual*		

Table 17. Default Client Framework Objects (continued)			
Step Attribute Path	**System Step Attribute**	**As condition for Event**	**feeds Dimension(s)**
`.sessions[0].message. performance.timing. connectEnd`	"Step - Performance Connect End" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message. performance.timing. connectStart`	"Step - Performance Connect Start" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message. performance.timing. domainLookupEnd`	"Step - Performance Dom Lookup End" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message. performance.timing. domainLookupStart`	"Step - Performance Dom Lookup Start" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message. performance.timing. domComplete`	"Step - Performance Dom Complete" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message. performance.timing. domContentLoadedEventEnd`	"Step - Performance Dom Content Loaded Event End" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message. performance.timing. domContent LoadedEventStart`	"Step - Performance Dom Loaded Event Start" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message. performance.timing. domInteractive`	"Step - Performance Dom Interactive" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message. performance.timing. domLoading`	"Step - Performance Dom Loading" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message. performance.timing. fetchStart`	"Step - Performance Fetch Start" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message. performance.timing. loadEventEnd`	"Step - Performance Load Event End" in the *IBM Tealeaf Event Manager Manual*		

Table 17. Default Client Framework Objects (continued)			
Step Attribute Path	**System Step Attribute**	**As condition for Event**	**feeds Dimension(s)**
`.sessions[0].message.performance.timing.navigationStart`	"Step - Performance Navigation Start" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.performance.timing.redirectEnd`	"Step - Performance Redirect End" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.performance.timing.redirectStart`	"Step - Performance Redirect Start" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.performance.timing.responseEnd`	"Step - Performance Response End" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.performance.timing.responseStart`	"Step - Performance Response Start" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.performance.timing.secureConnectionStart`	"Step - Performance Secure Connection Start" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.performance.timing.unloadEventEnd`	"Step - Performance Unload Event End" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.performance.timing.unloadEventStart`	"Step - Performance Unload Event Start" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.screenview.name`	"Step - ScreenView Name" in the *IBM Tealeaf Event Manager Manual*	"Step - ScreenView (BB)" in the *IBM Tealeaf Event Manager Manual*; "Step - ScreenView URL (BB)" in the *IBM Tealeaf Event Manager Manual*	"Step - ScreenView" in the *IBM Tealeaf Event Manager Manual*; "Step - ScreenView URL" in the *IBM Tealeaf Event Manager Manual*
`.sessions[0].message.screenview.renderTime`	"Step - ScreenView Render Time" in the *IBM Tealeaf Event Manager Manual*		

Table 17. Default Client Framework Objects (continued)

Step Attribute Path	System Step Attribute	As condition for Event	feeds Dimension(s)
`.sessions[0].message.screenview.type`	"Step - ScreenView Type" in the *IBM Tealeaf Event Manager Manual*	"Step - ScreenView (BB)" in the *IBM Tealeaf Event Manager Manual*; "Step - ScreenView URL (BB)" in the *IBM Tealeaf Event Manager Manual*	"Step - ScreenView" in the *IBM Tealeaf Event Manager Manual*; "Step - ScreenView URL" in the *IBM Tealeaf Event Manager Manual*
`.sessions[0].message.screenview.url`	"Step - ScreenView URL" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.screenviewOffset`	"Step - ScreenView Offset" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.currState.label`	"Step - Target Current® Label" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.currState.text`	"Step - Target Current Text" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.currState.value`	"Step - Target Current Value" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.dwell`	"Step - Target Dwell Time" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.id`	"Step - Target ID" in the *IBM Tealeaf Event Manager Manual*	"Step - Usability Target ID + Type (BB)" in the *IBM Tealeaf Event Manager Manual*	"Step - Target ID" in the *IBM Tealeaf Event Manager Manual*
`.sessions[0].message.target.idType`	"Step - Target ID Type" in the *IBM Tealeaf Event Manager Manual*	"Step - Usability Target ID + Type (BB)" in the *IBM Tealeaf Event Manager Manual*	"Step - Target ID" in the *IBM Tealeaf Event Manager Manual*
`.sessions[0].message.target.name`	"Step - Target Name" in the *IBM Tealeaf Event Manager Manual*		

Table 17. Default Client Framework Objects (continued)

Step Attribute Path	System Step Attribute	As condition for Event	feeds Dimension(s)
`.sessions[0].message.target.position.height`	"Step - Target Height" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.position.relXY`	"Step - Target Relative XY" in the *IBM Tealeaf Event Manager Manual*	"Step - Usability Click" in the *IBM Tealeaf Event Manager Manual*	"Step - Target Relative XY" in the *IBM Tealeaf Event Manager Manual*
`.sessions[0].message.target.position.width`	"Step - Target Width" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.prevState.label`	"Step - Target Prev Label" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.prevState.text`	"Step - Target Prev Text" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.prevState.value`	"Step - Target Previous Value" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.subType`	"Step - Target Subtype" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.tlType`	"Step - Target TL Type" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.type`	"Step - Target Type" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.visitedCount`	"Step - Target Visit Count" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].message.target.xPath`			
`.sessions[0].message.type`	"Step - Message Type" in the *IBM Tealeaf Event Manager Manual*		

Table 17. Default Client Framework Objects (continued)			
Step Attribute Path	**System Step Attribute**	**As condition for Event**	**feeds Dimension(s)**
`.sessions[0].startTime`	"Step - Session Start Time" in the *IBM Tealeaf Event Manager Manual*		
`.sessions[0].timezoneOffset`	"Step - Timezone Offset" in the *IBM Tealeaf Event Manager Manual*		

Tealeaf JSON schema - tlType

Different platforms use different names for objects, for example Android uses `button`, iOS uses `uibutton`, and UIC uses <button>. Tealeaf uses `tlType` internally to assign a single type to all objects that are the same type, regardless of how they are referred to on the platform that sent the message. For example, `ltType=button` means that in Tealeaf the object type is button, while on the platform that sent the message the control might be defined as a `uibutton`.

The `tlType` property is included for all controls that are monitored by each Tealeaf client framework. Whether an individual control is supported by one or all of the client frameworks, it can be referenced for eventing purposes through the `tlType` property. This table provides reference information about how `tlType` values map across all of the supported client frameworks. N/A indicates that the control is not supported in the client framework.

Table 18. Tealeaf JSON schema - tlType				
tlType	**UIC**	**Android**	**iOS**	**Description**
barButtonItem	N/A	N/A	UIBarButtonItem	A bar button item is a button that is specialized for placement on a UIToolbar or UINavigationBar object. It inherits basic button behavior from its abstract superclass, UIBarItem.
barItem	N/A	N/A	UIBarItem	UIBarItem is an abstract superclass for items added to a bar that appears at the bottom of the screen. Items on a bar behave in a way similar to buttons (instances of UIButton).
button	<button> or <input type="button">	Button	UIButton	Represents a push-button widget.

Table 18. Tealeaf JSON schema - tlType (continued)

tlType	UIC	Android	iOS	Description
button	N/A	Compound Button	N/A	A button with two states, checked and cleared.
button	<input type="button">	ImageButton	UIButton	Displays a button with an image (instead of text) that can be pressed or clicked by the user.
calendar	N/A	CalendarView	N/A	
canvas	<canvas>	view	UIView	
checkBox	<input type= "checkbox">	CheckBox	N/A	A check box is a specific type of two-states button that can be either checked or cleared.
command	<command>	N/A	N/A	The <command> tag defines a command (a radio button, a check box, or a command button) that the user can start.
datePicker	N/A	DatePicker	UIDatePicker	This class is a widget for selecting a date.
dialerFilter	N/A	DialerFilter	N/A	It converts letters to text according to old-fashioned phone keyboards (abc=2, def=3, etc)
gallery	N/A	Gallery	N/A	A view that shows items in a center-locked, horizontally scrolling list.

Table 18. Tealeaf JSON schema - tlType (continued)

tlType	UIC	Android	iOS	Description
gesture	N/A	android.gesture	UIGesture Recognizer	UIGesture Recognizer is an abstract base class for concrete gesture-recognizer classes. A gesture-recognizer object (or a gesture recognizer) decouples the logic for recognizing a gesture and acting on that recognition. When one of these objects recognizes a common gesture or, in some cases, a change in the gesture, it sends an action message to each designated target object.
gesture	N/A	android.gesture	UILongPress Gesture Recognizer	UILongPress Gesture Recognizer is a concrete subclass of UIGesture Recognizer that looks for long-press gestures. The user must press one or more fingers on a view for at least a specified period for the action message to be sent. In addition, the fingers may move only a specified distance for the gesture to be recognized; if they move beyond this limit the gesture fails.

Table 18. Tealeaf JSON schema - tlType (continued)				
tlType	**UIC**	**Android**	**iOS**	**Description**
gesture	N/A	android.gesture	UIPanGesture Recognizer	UIPanGesture Recognizer is a concrete subclass of UIGesture Recognizer that looks for panning (dragging) gestures. The user must be pressing one or more fingers on a view while they pan it. Clients implementing the action method for this gesture recognizer can ask it for the current translation and velocity of the gesture.
gesture	N/A	android.gesture	UIPinchGesture Recognizer	UIPinchGesture Recognizer is a concrete subclass of UIGesture Recognizer that looks for pinching gestures involving two touches. When the user moves the two fingers toward each other, the conventional meaning is zoom-out; when the user moves the two fingers away from each other, the conventional meaning is zoom-in.

Table 18. Tealeaf JSON schema - tlType (continued)				
tlType	**UIC**	**Android**	**iOS**	**Description**
gesture	N/A	android.gesture	UIRotation Gesture Recognizer	UIRotation Gesture Recognizer is a concrete subclass of UIGesture Recognizer that looks for rotation gestures involving two touches. When the user moves the fingers opposite each other in a circular motion, the underlying view should rotate in a corresponding direction and speed.
gesture	N/A	android.gesture	UISwipe Gesture Recognizer	UISwipe Gesture Recognizer is a concrete subclass of UIGesture Recognizer that looks for swiping gestures in one or more directions. A swipe is a discrete gesture, and thus the associated action message is sent only once per gesture.
gesture	N/A	android.gesture	UITap Gesture Recognizer	UITap Gesture Recognizer is a concrete subclass of UIGesture Recognizer that looks for single or multiple taps. For the gesture to be recognized, the specified number of fingers must tap the view a specified number of times.

Table 18. Tealeaf JSON schema - tlType (continued)

tlType	UIC	Android	iOS	Description
imagePicker	N/A	N/A	UIImage PickerController	The UIImage PickerController class manages customizable, system-supplied user interfaces for taking pictures and movies on supported devices, and for choosing saved images and movies for use in your app. An image picker controller manages user interactions and delivers the results of those interactions to a delegate object.
image Switcher	N/A	ImageSwitcher	N/A	It is a view useful to switch smoothly between two images and thus provides ways of transitioning from one to another through appropriate animations.
link	<a>	N/A	N/A	Defines a hyperlink
menu	<menu>	N/A	UIMenu Controller	The singleton UIMenu Controller instance presents the menu interface for the Cut, Copy, Paste, Select, Select All, and Delete commands.
menuItem	N/A	N/A	UIMenuItem	An instance of the UIMenuItem class represents a custom item in the editing menu managed by the UIMenu Controller object.

Table 18. Tealeaf JSON schema - tlType (continued)

tlType	UIC	Android	iOS	Description
navigation Bar	N/A	N/A	UINavigation Bar	The UINavigationBar class implements a control for navigating hierarchical content. It's a bar, typically displayed at the top of the screen, containing buttons for navigating up and down a hierarchy. The primary properties are a left (back) button, a center title, and an optional right button. You can specify custom views for each of these.
navigation Controller	N/A	N/A	UINavigation Controller	The UINavigation Controller class implements a specialized view controller that manages the navigation of hierarchical content. This class is not intended for subclassing. Instead, you use instances of it as is in situations where you want your application's user interface to reflect the hierarchical nature of your content. This navigation interface makes it possible to present your data efficiently and also makes it easier for the user to navigate that content.

Table 18. Tealeaf JSON schema - tlType (continued)				
tlType	**UIC**	**Android**	**iOS**	**Description**
navigationItem	N/A	N/A	UINavigation Item	The UINavigation Item class encapsulates information about a navigation item that is pushed on a UINavigation Bar object's stack. A navigation bar is a control that is used to navigate hierarchical content. A UINavigation Item specifies what is displayed on the navigation bar when it is the top item and also how it is represented when it is the back item.
numberPicker	N/A	NumberPicker	N/A	A widget that enables the user to select a number form a predefined range.
page	Page	Activity	UIView Controller	

Table 18. Tealeaf JSON schema - tlType (continued)

tlType	UIC	Android	iOS	Description
pageControl	N/A	N/A	UIPageControl	You use the UIPageControl class to create and manage page controls. A page control is a succession of dots that are centered in the control. Each dot corresponds to a page in the application's document (or other data-model entity), with the white dot indicating the currently viewed page. For example, When a user taps a page control to move to the next or previous page, the control sends the UIControlEvent ValueChanged event for handling by the delegate. The delegate can then evaluate the currentPage property to determine the page to display. The page control advances only one page in either direction.
quickContact Badge	N/A	QuickContact Badge	N/A	Widget that is used to show an image with the standard QuickContact badge and on-click behavior.
radioButton	<input type="radio">	RadioButton	N/A	A radio button is a two-states button that can be either checked or cleared.
radioButton	<input type="radio">	RadioGroup	N/A	This class is used to create a multiple-exclusion scope for a set of radio buttons.

Table 18. Tealeaf JSON schema - tlType (continued)				
tlType	**UIC**	**Android**	**iOS**	**Description**
scroll		Horizontal ScrollView	UIScrollView	Layout container for a view hierarchy that can be scrolled by the user, allowing it to be larger than the physical display.
scroll		Scroller	UIScrollView	This class encapsulates scrolling.
scroll		ScrollView	UIScrollView	Layout container for a view hierarchy that can be scrolled by the user, allowing it to be larger than the physical display.
searchBox	N/A	SearchView	UISearchBar	A widget that provides a user interface for the user to enter a search query and submit a request to a search provider.
selectList	<select>	AbsSpinner	UIPickerView, UITableViewCell	An abstract base class for spinner widgets.
selectList	<select>	Spinner	UIPickerView	A view that displays one child at a time and lets the user pick among them.
slider	<input type="range">	AbsSeekBar	UISlider	
slider	N/A	RatingBar	UISlider	A RatingBar is an extension of SeekBar and ProgressBar that shows a rating in stars.
slider	<input type="range">	SeekBar	UISlider	A SeekBar is an extension of ProgressBar that adds a draggable thumb.

Table 18. Tealeaf JSON schema - tlType (continued)				
tlType	**UIC**	**Android**	**iOS**	**Description**
stepper		N/A	UIStepper	A stepper displays two buttons, one with a minus (-) symbol and one with a plus (+) symbol. The bounding rectangle for a stepper matches that of a UISwitch object.
tabBar	N/A	TabWidget	UITabBar	The UITabBar class implements a control for selecting one of two or more buttons, called items. The most common use of a tab bar is to implement a modal interface where tapping an item changes the selection. Use a UIToolbar object if you want to momentarily highlight or not change the appearance of an item when tapped. The UITabBar class provides the ability for the user to customize the tab bar by reordering, removing, and adding items to the bar. You can use a tab bar delegate to augment this behavior.

Table 18. Tealeaf JSON schema - tlType (continued)

tlType	UIC	Android	iOS	Description
tabBarItem		TabHost.TabSpec	UITabBarItem	The UITabBarItem class implements an item on a tab bar, instances of the UITabBar class. A tab bar operates strictly in radio mode, where one item is selected at a time by tapping a tab bar item toggles the view above the tab bar. You can also specify a badge value on the tab bar item for adding more visual information, for example, the phone application uses a badge on the item to show the number of new messages. This class also provides a number of system defaults for creating items.
tabContainer	N/A	TabHost	UITabBar Controller	Container for a tabbed window view.
textBox	N/A	AutoComplete TextView	N/A	An editable text view that shows completion suggestions automatically while the user is typing.
textBox	N/A	Checked TextView	N/A	An extension to TextView that supports the Checkable interface.
textBox	<input>	EditText	UITextField	EditText is a thin veneer over TextView that configures itself to be editable.

Table 18. Tealeaf JSON schema - tlType (continued)				
tlType	**UIC**	**Android**	**iOS**	**Description**
textBox	N/A	MultiAuto CompleteText View	N/A	An editable text view, extending AutoComplete TextView, that can show completion suggestions for the substring of the text where the user is typing instead of necessarily for the entire thing.
textBox	<input type="text">	TextView	UITextField	Displays text to the user and optionally allows them to edit it.
textBox	<textarea>	EditText	UITextView	The UITextView class implements the behavior for a scrollable, multiline text region. The class supports the display of text using a custom font, color, and alignment and also supports text editing. You typically use a text view to display multiple lines of text, such as when displaying the body of a large text document.
timePicker	N/A	TimePicker	N/A	A view for selecting the time of day, in either 24 hour or AM/PM mode.
toggleButton	N/A	Switch	UISwitch	A Switch is a two-state toggle switch widget that can select between two options.
toggleButton	N/A	ToggleButton	UISwitch	Displays checked/ unchecked states as a button with a "light" indicator and by default that is accompanied with the text "ON" or "OFF".

Table 18. Tealeaf JSON schema - tlType (continued)				
tlType	**UIC**	**Android**	**iOS**	**Description**
webView	N/A	WebView	UIWebView	You use the UIWebView class to embed web content in your application. To do so, you simply create a UIWebView object, attach it to a window, and send it a request to load web content. You can also use this class to move back and forward in the history of webpages, and you can even set some web content properties program- matically.

Tealeaf JSON schema - tlEvent

Each Tealeaf client framework tracks user interface events, some of which are specific to the type of client application. Through the `tlEvent` property on the object controls, you can create one event to track all corresponding user interface events across all of the frameworks. In the table below, for each listed `tlType`, you can review the `tlEvent` reference, if applicable, and the corresponding object information for each supported client framework.

Event behavior

Some event behaviors apply to all client frameworks.

This table provides descriptions of the behaviors of each monitored tlEvent. These behaviors apply to all client frameworks.

Table 19. Event Behavior			
tlType	**tlEvent only on these Control objects**	**tl JSON Object type**	**Event Behavior**
actionSheet	buttonIndex	Control	Indicates which button user selected from action sheet
	showInView		When action sheet is shown in the view
	showFromTabBar		When action sheet is shown from tab bar
	showFromToolbar		When action sheet is shown from toolbar
alertView	alertShown	Control	When alert is shown
	buttonIndex		Indicates which button user selected from alert

Table 19. Event Behavior (continued)

tlType	tlEvent only on these Control objects	tl JSON Object type	Event Behavior
button	click	Control	When user clicks button
calendar	dateChange	Control	When user changes date on calendar
canvas	N/A	Control	This is the base class where you design the control. So you use the class that extends this class.
checkBox	click	Control	When check box is clicked
datePicker	dateChange	Control	When date is changed on datePicker
gallery	click	Control	When item is clicks gallery
imagePicker	Not implemented	Control	N/A
gesture	Not implemented	Control	N/A
gesture	on any frameworks		
gesture			
gesture			
gesture			
gesture			
gesture			
link	click	Control	When user clicks link
menu	Not implemented	Control	N/A
numberPicker	valueChange	Control	When user picks a number
page		ScreenView	ScreenView get created
			Called before page appears to user
	LOAD		Current page model that user views on screen which ScreenView object notifies when page goes to foreground
			Called after page is loaded into memory
			Called before page goes to background
	UNLOAD		Current page model that user views on screen which ScreenView object notifies when page goes to background

Table 19. Event Behavior (continued)

tlType	tlEvent only on these Control objects	tl JSON Object type	Event Behavior
			When page gets garbage that is collected
pageControl	Not implemented	Control	N/A
radioButton	click	Control	When user clicks a radio button
radioButton			
scroll	scrollChange	Control	When user scrolls page
scroll			
searchBox	valueChange	Control	When user clicks search button
selectList	valueChange	Control	When user selects an item from the list
selectList			
slider	valueChange	Control	When user stops sliding the control
slider			
slider			
stepper	Not implemented		N/A
tabContainer	tabChange	Control	When user selects a tab
textBox	textChange	Control	When user does a focus out of a text box
textBox			
textBox			
textBox			
textBox			
textBox			
timePicker	timeChange	Control	When user changes time
toggleButton	click	Control	When user clicks a toggle button
toggleButton			

UI Capture

There are several events that are unique to UI Capture.

This table lists and describes the events for UI Capture:

Table 20. UI Capture				
tlType	**tlEvent only on these Control objects**	**tl JSON Object type**	**UIC**	**UIC Event - See Below at TL table for HTML**
button	click	Control	<button>	click, blur
button	click	Control	<input>	click, blur
checkBox	change	Control	<input>	change, blur
link	click	Control	<a>	click
page		ScreenView	Page	
	LOAD			load
	UNLOAD			unload
radio Button	change	Control	<input>	change, blur
scroll	scrollChange	Control		
selectList	valueChange	Control	<select>	change, blur
textBox	textChange	Control	<input>	change, blur
textBox	textChange	Control	<textarea>	change, blur

Android

There are several events that are unique to the Android framework.

This table lists the events that are unique to the Android framework:

Table 21. Android					
tlType	**tlEvent only on these Control objects**	**tl JSON Object type**	**Android**	**Android Listener**	**Android Event**
actionSheet	buttonIndex	Control	N/A	N/A	N/A
	showInView				
	showFrom TabBar				
	showFrom Toolbar				
alertView	alertShown	Control	N/A	N/A	N/A
	buttonIndex				
button	click	Control	Button	View.On ClickListener	onClick
button			Compound Button		
button			ImageButton		
calendar	dateChange	Control	CalendarView	CalendarView. OnDate Change Listener	onSelected DayChange

Table 21. Android (continued)

tlType	tlEvent only on these Control objects	tl JSON Object type	Android	Android Listener	Android Event
canvas	N/A	Control	view	On the class it self	onFinish Inflate
					onFocus Changed
					onLayout
					onMeasure
					onOver Scrolled
					onRestore InstanceState
					onScroll Changed
					onSetAlpha
					onSize Changed
					onVisibility Changed
					onWindow Visibility Changed
checkBox	click	Control	CheckBox	View.On ClickListener	onClick
datePicker	dateChange	Control	DatePicker	DatePicker. OnDate Changed Listener	onDate Changed
gallery	click	Control	Gallery	AdapterView. OnItem ClickListener	onItemClick
image Picker	Not implemented	Control	N/A		
gesture	Not implemented	Control	android. gesture	Gesture Detector. OnGesture Listener	onDown
gesture	on any frameworks		android. gesture		onFling
gesture			android. gesture		onLongPress
gesture			android. gesture		onScroll
gesture			android. gesture		onShowPress
gesture			android. gesture		onSingle TapUp
gesture			android. gesture		

Table 21. Android (continued)

tlType	tlEvent only on these Control objects	tl JSON Object type	Android	Android Listener	Android Event
link	click	Control	N/A		
menu	Not implemented	Control	N/A		
number Picker	valueChange	Control	NumberPicker	Number Picker.On ValueChange Listener	onValue Change
page		ScreenView	Activity	On the class it self	onCreate
	LOAD				onResume
	UNLOAD				onPause
					onDestroy
page Control	Not implemented	Control	N/A		
radio Button	click	Control	RadioButton	View.On ClickListener	onClick
radio Button			RadioGroup	RadioGroup. OnChecked Change Listener	onChecked Changed
scroll	scrollChange	Control	Horizontal ScrollView	You have to extend Horizontal ScrollView	onScroll Changed
scroll			ScrollView	You have to extend ScrollView	onScroll Changed
searchBox	valueChange	Control	SearchView	SearchView Compat.On QueryText Listener Compat	onQuery TextSubmit
selectList	valueChange	Control	AbsSpinner	AdapterView. OnItem Selected Listener	onItem Selected
selectList			Spinner		
slider	valueChange	Control	AbsSeekBar	SeekBar. OnSeekBar Change Listener	onStop TrackingTouch
slider			SeekBar		
slider			RatingBar	RatingBar. OnRatingBar Change Listener	onRating Changed

Table 21. Android (continued)					
tlType	**tlEvent only on these Control objects**	**tl JSON Object type**	**Android**	**Android Listener**	**Android Event**
stepper	Not implemented		N/A		
tab Container	tabChange	Control	TabHost	TabHost. OnTab Change Listener	onTab Changed
textBox	textChange	Control	AutoComplete TextView	View.OnFocus Change Listener	onFocus Change
textBox			Checked TextView		
textBox			EditText		
textBox			MultiAuto Complete TextView		
textBox			TextView		
textBox			EditText		
timePicker	timeChange	Control	TimePicker	TimePicker. OnTime Changed Listener	onTime Changed
toggle Button	click	Control	Switch	View. OnClick Listener	onClick
toggle Button			ToggleButton		

iOS

There are several events that are unique to the iOS framework.

This table lists the events that are unique to the iOS framework:

Table 22. iOS					
tlType	**tlEvent only on these Control objects**	**tl JSON Object type**	**iOS**	**iOS Event**	**iOS TL Short Name**
action Sheet	buttonIndex	Control	UIActionSheet		clickedButton AtIndex
	showInView			Not implemented	actionSheet ShowInView
	showFrom TabBar			Not implemented	actionSheet ShowFrom TabBar
	showFrom Toolbar			Not implemented	actionSheet ShowFrom Toolbar

Table 22. iOS (continued)

tlType	tlEvent only on these Control objects	tl JSON Object type	iOS	iOS Event	iOS TL Short Name
alertView	alertShown	Control	UIAlertView	Not implemented	alertView Shown
	buttonIndex				clickedButton AtIndex
button	click	Control	UIButton		ButtonTouch UpInside
button			N/A		
button			UIButton		
calendar	dateChange	Control	N/A	N/A	N/A
canvas	N/A	Control	UIView		
checkBox	click	Control	N/A	N/A	N/A
datePicker	dateChange	Control	UIDatePicker	Not implemented	Not implemented
gallery	click	Control	N/A	N/A	N/A
image Picker	Not implemented	Control	UIImage Picker Controller	Not implemented	Not implemented
gesture	Not implemented	Control	UIGesture Recognizer		
gesture	on any frameworks		UILongPress Gesture Recognizer		
gesture			UIPan Gesture Recognizer		
gesture			UIPinch Gesture Recognizer		
gesture			UIRotation Gesture Recognizer		
gesture			UISwipe Gesture Recognizer		
gesture			UITap Gesture Recognizer		
link	click	Control	N/A	N/A	N/A
menu	Not implemented	Control	UIMenu Controller	Not implemented	Not implemented
number Picker	valueChange	Control	N/A	N/A	N/A

Table 22. iOS (continued)

tlType	tlEvent only on these Control objects	tl JSON Object type	iOS	iOS Event	iOS TL Short Name
page		ScreenView	UIView Controller		View Controller InitWith NibNamed
					View Controller InitWith Coder
					View Controller ViewWill Appear
	LOAD				View Controller ViewDid Appear
					View Controller ViewDid Load
					View Controller ViewWill Disappear
	UNLOAD				View Controller ViewDid Disappear
page Control	Not implemented	Control	UIPageControl	N/A	N/A
radio Button	click	Control	N/A	N/A	N/A
radio Button			N/A		
scroll	scrollChange	Control	UIScrollView	Not implemented	Not implemented
scroll			UIScrollView		
searchBox	valueChange	Control	UISearchBar	Not implemented	Not implemented
selectList	valueChange	Control	UIPickerView	Not implemented	Not implemented
selectList			UIPickerView		
slider	valueChange	Control	UISlider	No event name but we are handling clicks	No event name but we are handling clicks
slider			UISlider		
slider			UISlider		
stepper	Not implemented		UIStepper		
tab Container	tabChange	Control	UITabBar Controller	Not implemented	Not implemented

Table 22. iOS (continued)

tlType	tlEvent only on these Control objects	tl JSON Object type	iOS	iOS Event	iOS TL Short Name
textBox	textChange	Control	N/A		TextField DidBegin Editing
textBox			N/A		TextField DidEnd Editing
textBox			UITextField		SecureText FieldDid Begin Editing
textBox			N/A		SecureText FieldDid EndEditing
textBox			UITextField		
textBox			UITextView		TextView DidBegin Editing
					TextView DidEnd Editing
					SecureText ViewDid BeginEditing
					SecureText ViewDid EndEditing
timePicker	timeChange	Control	N/A	N/A	N/A
toggle Button	click	Control	UISwitch	No event name but we are handling clicks	No event name but we are handling clicks
toggle Button			UISwitch		

Tealeaf JSON Schema - Values for Controls

Objects from different client frameworks have different names. Objects of the same type are mapped internally to a `tlType`, so that an object of `tlType=button` is a button, no matter what the client framework calls a button.

For each `tlType` entry, you can see how it is supported in each client framework and the meaning of the value(s) as tracked in Tealeaf:

Table 23. Tealeaf JSON Schema - Values for Controls

tlType	UIC	iOS	Android	Android API Level	Android Instrument	Key	Value
link	<a>	N/A	N/A			text	Text being displayed between tag
						url	Url of hyperlink
slider	<input>	UISlider	AbsSeekBar	1	abstract class	maxValue	Maximum slider value
						value	Current slider value
selectList	<select>	UIPickerView	AbsSpinner	1	abstract class	text	Text being selected **Note:** Not currently supported in iOS Logging Framework.
textBox	N/A	N/A	AutoComplete TextView	1		text	
button	<button>	UIButton	Button	1		text	Text typed in text box
calendar	N/A	N/A	CalendarView	11		date	Date selected on calendar
checkBox	<input>	N/A	CheckBox	1		text	Text of checked item
textBox	N/A	N/A	CheckedTextView	1		text	Text of checked text
button	N/A	N/A	Compound Button	1	abstract class	text	Text of the label of the button
command	<command>	N/A	N/A			text	To be determined it, we currently do not support this control
datePicker	N/A	UIDatePicker	DatePicker	1		date	Date being picked
dialerFilter	N/A	N/A	DialerFilter	1		text	Text that was typed in dialer filter after being processed.
textBox	<input>	UITextField	EditText	1		text	Text typed in text box
gallery	N/A	N/A	Gallery	1		index	Index of item selected

Table 23. Tealeaf JSON Schema - Values for Controls (continued)

tlType	UIC	iOS	Android	Android API Level	Android Instrument	Key	Value
scroll	N/A	N/A	Horizontal ScrollView	3		x	Value of the scroll
						y	Value of the scroll
button	<input>	UIButton	ImageButton	1		imageName	Name of image being displayed as a button
textBox	N/A	N/A	MultiAutoComplete TextView	1		text	Text typed in text box
numberPicker	N/A	N/A	NumberPicker	11		text	Selected number
radioButton	<input>	N/A	RadioButton	1		text	Text of the value selected
radioButton	<input>	N/A	RadioGroup	1		text	Text of the value selected
slider	<input>	UISlider	RatingBar	1		maxValue	Maximum slider value
						value	Current slider value
scroll		UIScrollView	Scroller	1	not a view, computing in backend	x	Value of the scroll
						y	Value of the scroll
scroll		UIScrollView	ScrollView	1		x	Value of the scroll
						y	Value of the scroll
searchBox	N/A	UISearchBar	SearchView	11		text	Text being searched
slider	<input>	UISlider	SeekBar	1		maxValue	Maximum slider value
						value	Current slider value
selectList	<select>	UIPickerView	Spinner	1		text	Text being selected
toggleButton	N/A	UISwitch	Switch	14		isOn	true or false value if it is on
tabContainer	N/A	UITabBarController	TabHost	1		text	Selected tab

Table 23. Tealeaf JSON Schema - Values for Controls (continued)

tlType	UIC	iOS	Android	Android API Level	Android Instrument	Key	Value
textBox	<input>	UITextField	TextView	1		text	Text typed in text box
timePicker	N/A	N/A	TimePicker	1		time	Time that was picked
toggleButton	N/A	UISwitch	ToggleButton	1		isToggled	true or false value if it is toggled
barButton Item	N/A	UIBar ButtonItem	N/A			text	Title of button bar selected item
gesture	N/A	UIGesture Recognizer	android.gesture	4		N/A	Currently not being tracked
image Picker	N/A	UIImagePicker Controller	N/A			text	Name of image being displayed
gesture	N/A	UILongPress GestureRecognizer	android.gesture	4		N/A	Currently not being tracked
menuItem	N/A	UIMenuItem	N/A			text	Title of menu item selected
navigationItem	N/A	UINavigationItem	N/A			text	Title of item selected
navigationItem	N/A	UIPageControl	N/A			page	Current page being displayed
gesture	N/A	UIPanGesture Recognizer	android.gesture	4		N/A	Currently not being tracked
gesture	N/A	UIPinchGesture Recognizer	android.gesture	4		N/A	Currently not being tracked
gesture	N/A	UIRotationGesture Recognizer	android.gesture	4		N/A	Currently not being tracked
stepper		UIStepper	N/A			value	Value of the stepper
gesture	N/A	UISwipeGesture Recognizer	android.gesture	4		N/A	Currently not being tracked
stepper	N/A	UITabBar	TabWidget	1		text	Title of seleted tab item

Table 23. Tealeaf JSON Schema - Values for Controls (continued)

tlType	UIC	iOS	Android	Android API Level	Android Instrument	Key	Value
gesture	N/A	UITapGesture Recognizer	android.gesture	4		N/A	Currently not being tracked
textBox	<text- area>	UITextView	EditText	1		text	Text typed in text box
actionSheet		ActionSheet				text	Text of the button clicked
						button Index	Index of the button clicked
alertView		AlertView				text	Text of the button clicked
						button Index	Index of the button clicked

IBM Tealeaf documentation and help

IBM Tealeaf provides documentation and help for users, developers, and administrators.

Viewing product documentation

All IBM Tealeaf product documentation is available at the following website:

Tealeaf Customer Experience Support

Use the information in the following table to view the product documentation for IBM Tealeaf:

Table 24. Getting help

To view...	**Do this...**
Product documentation	On the IBM Tealeaf portal, go to **?** > **Product Documentation**.
IBM Tealeaf Knowledge Center	On the IBM Tealeaf portal, go to **?** > **Product Documentation** and select *IBM Tealeaf Customer Experience in the ExperienceOne Knowledge Center*.
Help for a page on the IBM Tealeaf Portal	On the IBM Tealeaf portal, go to **?** > **Help for This Page**.
Help for IBM Tealeaf CX PCA	On the IBM Tealeaf CX PCA web interface, select **Guide** to access the *IBM Tealeaf CX PCA Manual*.

Available documents for IBM Tealeaf products

The following table is a list of available documents for all IBM Tealeaf products:

Table 25. Available documentation for IBM Tealeaf products

IBM Tealeaf products	**Available documents**
IBM Tealeaf CX	• *IBM Tealeaf Customer Experience Overview Guide* • *IBM Tealeaf CX Client Framework Data Integration Guide* • *IBM Tealeaf CX Configuration Manual* • *IBM Tealeaf CX Cookie Injector Manual* • *IBM Tealeaf CX Databases Guide* • *IBM Tealeaf CX Event Manager Manual* • *IBM Tealeaf CX Glossary* • *IBM Tealeaf CX Installation Manual* • *IBM Tealeaf CX PCA Manual* • *IBM Tealeaf CX PCA Release Notes*

Table 25. Available documentation for IBM Tealeaf products (continued)	
IBM Tealeaf products	**Available documents**
IBM Tealeaf CX	• *IBM Tealeaf CX RealiTea Viewer Client Side Capture Manual* • *IBM Tealeaf CX RealiTea Viewer User Manual* • *IBM Tealeaf CX Release Notes* • *IBM Tealeaf CX Release Upgrade Manual* • *IBM Tealeaf CX Support Troubleshooting FAQ* • *IBM Tealeaf CX Troubleshooting Guide* • *IBM Tealeaf CX UI Capture j2 Guide* • *IBM Tealeaf CX UI Capture j2 Release Notes*
IBM Tealeaf cxImpact	• *IBM Tealeaf cxImpact Administration Manual* • *IBM Tealeaf cxImpact User Manual* • *IBM Tealeaf cxImpact Reporting Guide*
IBM Tealeaf cxConnect	• *IBM Tealeaf cxConnect for Data Analysis Administration Manual* • *IBM Tealeaf cxConnect for Voice of Customer Administration Manual* • *IBM Tealeaf cxConnect for Web Analytics Administration Manual*
IBM Tealeaf cxOverstat	*IBM Tealeaf cxOverstat User Manual*
IBM Tealeaf cxReveal	• *IBM Tealeaf cxReveal Administration Manual* • *IBM Tealeaf cxReveal API Guide* • *IBM Tealeaf cxReveal User Manual*
IBM Tealeaf cxVerify	• *IBM Tealeaf cxVerify Installation Guide* • *IBM Tealeaf cxVerify User's Guide*
IBM Tealeaf cxView	*IBM Tealeaf cxView User's Guide*
IBM Tealeaf CX Mobile	• *IBM Tealeaf CX Mobile Android Logging Framework Guide* • *IBM Tealeaf Android Logging Framework Release Notes* • *IBM Tealeaf CX Mobile Administration Manual* • *IBM Tealeaf CX Mobile User Manual* • *IBM Tealeaf CX Mobile iOS Logging Framework Guide* • *IBM Tealeaf iOS Logging Framework Release Notes*

Index