Session ID API for 3rd party client-side scripts

## Revision History

|  |  |
| --- | --- |
| 0.1 | Initial draft circulated for review. |
| 0.2 | Updated assumptions and solution to include the query parameter option.  Elaborate on the downside of Approach #2 per feedback from Steve W. |
| 0.3 | Updated assumption 3 to elaborate on the page context bit. |

## I. Problem Statement

3rd party services attempting to integrate with Tealeaf need a mechanism to identify the session cookie used by Tealeaf.

## II. Assumptions

1. The session id is derived from a single cookie or query string parameter.
2. The API provides the cookie or query parameter value used to derive the session ID and not necessarily the actual session ID. The exception is when TLTSID is used. A MD5 hash on the value is required to derive the actual session ID. This operation is outside the scope of this API.
3. The 3rd party JavaScript is executing in the same page domain as the Tealeaf UI Capture JavaScript. Cross-domain access is not supported. The script does not have to physically reside on the same server as the page itself but it does need to be directly included by the page in order to operate in the same page domain and context.
4. Tealeaf UI Capture JavaScript will be configured with the cookie name or query parameter that is being used to sessionize the hits. In the absence of explicit configuration, “TLTSID” will be the default cookie name.
5. The session cookie value will not change during the lifetime of the page.

## III. Design Approach/Architecture Overview

There are two API proposals which are described below:

1. API returning an opaque handle (string)

string TLT.getSessionData(void);

The format of the returned string is as follows:

*SessionCookieName=SessionCookieValue*

or

*SessionQueryName=SessionQueryValue\**

e.g.

TLTSID=1909E4406FAB106F070186B980A85F96

or

jsessionid=AFqX4MRMG9NaFfHu0

\*Note: For brevity henceforth the SessionCookie prefix will be used to mean both cookie and query string.

The advantage of this approach is the relatively minimal overhead required for the management of the handle.

The disadvantage of this approach is the lack of extensibility.

Extensibility can be provided by further segmenting the opaque string into “;” separated parts. If additional data needs to be shared then it will be serialized into appropriate name-value pairs and appended to the opaque handle.

e.g. The following opaque handles contain the session cookie and user id information.

TLTSID=1909E4406FAB106F070186B980A85F96;userid=jdoe

or

jsessionid=AFqX4MRMG9NaFfHu0;userid=jdoe

1. API returning a JSON object with name-value pairs.

Object TLT.getSessionData(void);

The format of the returned object is as follows:

{

tltSCN: *SessionCookieName,*

tltSCV: *SessionCookieValue*

}

e.g.

{

tltSCN: "TLTSID",

tltSCV: "1909E4406FAB106F070186B980A85F96"

}

or

{

tltSCN: "jsessionid",

tltSCV: "AFqX4MRMG9NaFfHu0"

}

The advantage of this approach is the flexibility and ease of extensibility. Additional name-value pairs can be added to the returned object without adversely impacting legacy implementations.

The disadvantage of this approach is the **relatively** complex management of the returned data object. For example, a string is a primitive data type that is portable across almost all platforms but objects are not. By itself, this is not a significant downside.

## IV. Solution

The implementation for the API will be as follows:

1. If the session cookie has been previously stored then skip to step 5.
2. Read the core configuration and examine it for the Tealeaf session cookie name or query parameter to use. If one doesn’t exist then use TLTSID as the session cookie name.
3. If a query parameter is configured, read and parse the query string. Find the query parameter. Go to step 5.
4. Else if a cookie name is configured, read and parse the cookies on the document object. Find the session cookie.
5. Store the value for future invocations of the API (if any)
6. If no query parameter or session cookie is found then return null.
7. Return the value using the template of the chosen approach:

i.e. either a string: "SessionCookieName=SessionCookieValue"

or an object:

{

tltSCN: *SessionCookieName,*

tltSCV: *SessionCookieValue*

}

## V. Dependencies (People, Products, etc.)

NA

## VI. Non-functional Attributes

NA

## VII. i18n (Internationalization) and L10n (Localization)

NA

## VIII. Test Strategy

Automated unit tests will be built and integrated into the existing UIC unit testing framework.

## IX. Existing Jiras

DAR-34

API Documentation needs to be updated.