Unity SDK

You can send data from your Unity application to Treasure Data, using our Unity SDK library. In this way, you don't have to install anything on your server-side to track the mobile app activities.

This topic includes:

GDPR Compliance

To support compliance with national and global data privacy requirements such as the European General Data Privacy Regulation, our SDK provides methods that control the collection and tracking of personal data and metadata in applications and websites. When your company defines data privacy policies around personal data, you can use these methods in your code to implement default data collection behaviors, and add controls for individuals to use to manage data collection and privacy themselves.

Customers of Treasure Data must ensure that their usage of the SDK, including its use that collects personal data, complies with the legal agreement that governs access to and use of the Treasure Data service, including specifically:

Treasure Data	URL
Terms of Service	https://www.treasuredata.com/terms/
Privacy Policy	https://www.treasuredata.com/privacy/
Privacy Statement for Customer Data	https://www.treasuredata.com/td-downloads/Privacy-Statement-for-Customer-Data.pdf

Prerequisites

- Basic knowledge of Unity Development
- Basic knowledge of Treasure Data

Our Unity SDK is available through Github. Check the repository to ensure that you have the most recent SDK.

- https://github.com/treasure-data/td-unity-sdk-package

Install the Library

1. Download the most recent version of our Unity package.
2. Import it into your Unity project using Assets > Import Package > Custom Package.

For iOS Application development

On Xcode, you must complete the following steps:

1. In Build Phases > Link Binary With Libraries.
2. Add libz.tbd
3. In Build Phases > Compile Sources.
4. Add -fno-objc-arc compile flag to NativePlugin.mm

Initialize the Library

Next, initialize the library in your app as follows.

```
public class MyTreasureDataPlugin : MonoBehaviour {
#if UNITY_IPHONE || UNITY_ANDROID
  [RuntimeInitializeOnLoadMethod(RuntimeInitializeLoadType.BeforeSceneLoad)]
  static void OnRuntimeInitialization() {
    TreasureData.InitializeApiEndpoint("https://in.treasuredata.com");
    TreasureData.InitializeApiKey("YOUR_API_KEY");
    ...
    /*
     * This is optional, but you can encrypt the buffered data on mobile devices.
     * You can prevent people from checking the buffered events on the disk.
     */
    // TreasureData.InitializeEncryptionKey("RANDOM_STRING_TO_ENCRYPT_DATA");
  }

  void OnApplicationPause(bool pauseStatus) {
    // Make an open request whenever app is resumed
    if (!pauseStatus) {
      TreasureData.Instance.UploadEvents(
        delegate() { Debug.LogWarning ("UploadEvents Success!!! "); },
        delegate(string errorCode, string errorMsg) { Debug.LogWarning ("UploadEvents Error!!! errorCode=" +
errorCode + ", errorMsg=" + errorMsg); }
      );
    }
  }
#endif
}
```

The API key can be retrieved from the TD Console. It's recommended that you use the write-only API key for SDKs. Let us know if you're having any build issues.

When to upload and how often to upload buffered events depends on the characteristics of your application. Good times to upload include:

- When the current screen is closing or moving to background
- When closing the application

Send Events to the Cloud

Next, call the `AddEvent()` function at the appropriate time within your applications. The following example shows an event is sent to the table `table_b` within the database `database_a`.

```
Dictionary<string, object> ev = new Dictionary<string, object>();
ev["str"] = "strstr";
ev["int"] = 12345;
ev["long"] = 12345678912345678;
ev["float"] = 12.345;
ev["double"] = 12.3459832987654;
ev["bool"] = true;
TreasureData.Instance.AddEvent("database_a", "table_b", ev);
```

IP whitelist won't be applied to any import from iOS or Android SDK. Also we've seen a lot of cases where a lot of iOS or Android devices have an invalid timestamp (like 1970/01/01), so we're currently ignoring the log which has a timestamp older than 7 days, and newer than 3 days ahead.

Tracking Application Lifecycle Events

Optionally, you can enable the Unity SDK to automatically capture app lifecycle events (disabled by default). You must explicitly enable this option. You can set the target table through `DefaultTable`:

```
[RuntimeInitializeOnLoadMethod(RuntimeInitializeLoadType.BeforeSceneLoad)]
static void OnRuntimeInitialization() {
  // TreasureData client setup...
  TreasureData.DefaultTable = "app_lifecycles";
  TreasureData.Instance.EnableAppLifecycleEvent();
}
```

There are 3 kinds of event that are tracked automatically:

- Application Open
- Install
- Update

These events are captured along with relevant metadata depends on the specific type of event:

Application Open

```
{
    "td_unity_event": "TD_UNITY_APP_OPEN",
    "td_app_ver": "1.0",
    ...
}
```

Application Install

```
{
    "td_unity_event": "TD_UNITY_APP_INSTALL",
    "td_app_ver": "1.0",
    ...
}
```

Application Update

```
{
    "td_unity_event": "TD_UNITY_APP_UPDATE",
    "td_app_ver": "1.1",
    "td_app_previous_ver": "1.0",
    ...
}
```

Tracking In-App Purchase Events

If enabled, Treasure Data SDK can automatically track IAP events without having to listen and call `addEvent` yourself. Enable this tracking feature by:

```
TreasureData.sharedInstance.enableInAppPurchaseEvent();
```

Although always disabled by default (unlike the other options: app lifecycle events or custom events, this choice is not persisted), you could explicitly disable this feature by:

```
TreasureData.sharedInstance.disableInAppPurchaseEvent();
```

Depends on the native platform in which the game is running on, the event's schema could be different. On Android, the IAP event's columns consist of:

- td_android_event
- td_iap_product_id
- td_iap_order_id
- td_iap_product_price
- td_iap_quantity
- td_iap_product_price_amount_micros
- td_iap_product_currency
- td_iap_purchase_time
- td_iap_purchase_token
- td_iap_purchase_state
- td_iap_purchase_developer_payload
- td_iap_product_type
- td_iap_product_title
- td_iap_product_description
- td_iap_package_name
- td_iap_subs_auto_renewing
- td_iap_subs_status
- td_iap_subs_period
- td_iap_free_trial_period

- td_iap_intro_price_period
- td_iap_intro_price_cycless
- td_iap_intro_price_amount_micros

While on iOS:

- td_ios_event
- td_iap_transaction_identifier
- td_iap_transaction_date
- td_iap_quantity
- td_iap_product_identifier
- td_iap_product_price
- td_iap_product_localized_title
- td_iap_product_localized_description
- td_iap_product_currency_code

For further details of these columns' value see:

- Treasure Data Android SDK - IAP Tracking
- Treasure Data iOS SDK - IAP Tracking

Opt-Out

Respecting your user's privacy is critical to any business. Treasure Data SDK can opt-out all events tracking for a particular device and de-identify users by resetting (or disabling completely) the `td_uuid.` You can change the `td_uuid.` to a different id for all subsequent events:

```
TreasureData.Instance.disableCustomEvent();         // disable your own events, ones that collect through
manual calls to addEvent...
TreasureData.Instance.disableAppLifecycleEvent();   // disable the auto-collected app lifecycle events
TreasureData.Instance.disableInAppPurchaseEvent();  // disable the auto-collected in-app purchase events
```

Unlike other option flags, enable/disableCustomEvent and enable/disableAppLifecycleEvent are persistent settings, which means that the settings survive across app launches. It is important to use the disableCustomEvent and disableAppLifecycleEvent calls whenever your users indicate that they don't want to be tracked. It is not necessary to call the options on every Treasure Data client setup.

```
TreasureData.Instance.resetUniqId();
TreasureData.Instance.disableAppendUniqId();

// temporary configuration, call Treasure Data for client setup
```

`resetUniqId` also adds an audit event to the `DefaultTable`

```
{
    "td_unity_event": "forget_device_uuid",
    "td_uuid": <old_uuid>,
    <configured_additional_parameters...>
}
```

Retry Uploading and Deduplication

The SDK imports events in one style with the combination of these features:

- This SDK keeps buffered events by adding unique keys and retries to upload them until confirming the events are uploaded and stored on the server-side (at least once)
- The server side remembers the unique keys of all events within the past 1 hour by default and can prevent duplicate imports.

Deduplication is a best effort system that identifies a duplicate record if a record with the same identifier is seen in the same dataset, within the last hour at most or within the last 4096 records, whichever comes first.