PayPal Magnes

Overview

This tutorial guides you to the steps required to integrate PayPal Magnes code into your **mobile** web page after you have activated PayPal as a payment method with Verifone. For the desktop site web page, please refer to PayPal Fraudnet.

PayPal Magnes accesses iOS and Android data for PayPal Risk Services to perform early risk identification and mitigation.

Magnes implementation

Magnes is integrated within the client app. It collects device-related information required for risk management. Magnes sends the information to PayPal Risk Services, which performs risk adjudication, thus providing a better customer experience.

Magnes payment flow

1. Generate a Tracking ID / AppGuid. Refer PayPal Risk Analysis for this.
2. Add required Android permissions or iOS frameworks.
3. The mobile app sets up Magnes.
4. Magnes accepts the Tracking ID passed in and sends it to the app.
5. Magnes collects and submits a payload of key device information along with the Tracking ID to PayPal Risk Services.
6. During transactions, the mobile app sends the transaction, along with the Tracking ID, to the merchant server.
7. The merchant server includes the `PayPalFraudId` header with the Tracking ID as the value in the call to the Verifone transaction API calls.
8. PayPal Risk Services utilizes the payload data to perform risk management for the transaction and reduce friction.

Magnes does not make payment or risk decisions, it only provides data to PayPal Risk Services to facilitate risk management.

Integrate Magnes

The Magnes library provides a set of APIs that host apps can invoke to use Magnes. You can integrate Magnes into these operating systems:

Android integration

Before you begin, the following table lists prerequisite information for integrating Magnes on the Android platform:

Prerequisite	Description
OS version	Use the latest Android version, if possible
Software configuration	The central class for Magnes is `lib.android.paypal.com.magnessdk.MagnesSDK`. This class provides single access for the host app.

Adding permissions

Before initializing the Magnes library, you must add permissions and metadata settings in the app's manifest file so Magnes can access essential mobile data for risk assessment.

Add the following in `AndroidManifest.XML`:

```
<manifest>
<!-- following permissions are optional -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<!-- for reading GSF ID -->
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
</manifest>
```

Setting up Magnes

Magnes must be set up at app startup. The `setUp` call is executed only once per lifecycle of a Magnes Singleton Instance. The Setup parameters and methods listed below are optional. You can set Magnes to use either a simple or advanced setup. The advanced setup gives more customizability and the option to pass in more parameters. Most developers will opt for a simple setup. Use the parameters with which you are familiar, otherwise build a default setup without using any parameter.

Magnes setup – simple

```
MagnesSettings magnesSettings = new MagnesSettings.Builder(context).build();
MagnesSDK.getInstance().setUp(magnesSettings);
```

Magnes setup – advanced

```
magnesSettings = new MagnesSettings.Builder(@NonNull Context context)
.setAppGuid(@Size(max = APPGUID_MAXLENGTH) String appGuid)
.setMagnesSource(@MagnesSource.SourceFlow int sourceFlow)
.setNotificationToken(String notificationToken)
.setMagnesNetworkingFactory
(MagnesNetworkingFactoryImpl magnesNetworkingFactoryImpl)
.enableNetworkOnCallerThread(boolean networkOnCallerThread)
.disableRemoteConfig(boolean disableRemoteConfig)
.build();
MagnesSDK.getInstance().setUp(magnesSettings);
```

Setup parameters and methods

Parameter or Method	Data Type	Description

`context`	Context	Android application context
`setAppGuid()`	String	Sets an application's globally unique identifier, which identifies the merchant application that sets up Magnes on the mobile device. If the merchant app does not pass an `AppGuid`, Magnes creates one to identify the app. If the app is installed or reinstalled, it receives a new `AppGuid`. Maximum length: 36 characters.
`SetMagnesSource()`	IntDef	Integration defined sources: `MagnesSource.PAYPAL`, `MagnesSource.EBAY`, and `MagnesSource.BRAINTREE`. If none of these apply to your integration, either use `MagnesSource.DEFAULT` or do not set a source.
`disableRemoteConfig()`	Boolean	Toggle to trigger a remote configuration networking call. It triggers once per application lifecycle. The default is `false`.
`enableNetworkOnCallerThread()`	Boolean	Enable a network call on the current caller thread. **Note:** It might throw a `NetworkOnMainThreadException` if your caller thread is the main thread.
`setMagnesNetworking Factory()`	`MagnesNetworkingFactoryImpl`	Magnes uses this custom networking factory instead of the Magnes default one to trigger the networking call. You must implement `MagnesNetworking` and `MagnesNetworkingFactory`.

iOS Swift SDK Integration

Before you begin, the following table lists prerequisite information for integrating Magnes on the iOS platform:

Prerequisite	Description
OS version	Using the latest iOS version is recommended
Software configuration	You can use the Magnes iOS Swift library for apps written in the Swift language

Prerequisite	Description
`MagnesSDK`	`MagnesSDK` is the central class for the Swift library. It provides singleton access for the host app.

Designate frameworks

Before you initialize Magnes, you must set several frameworks as **required** so Magnes can access essential mobile data for risk assessment. For example, by setting `CoreLocation.framework` as required, Magnes can access the current location of the mobile device.

Set the following frameworks as **required**.

- `Security.framework`
- `CFNetwork.framework`
- `SystemConfiguration.framework`
- `MessageUI.framework`
- `CoreLocation.framework`
- `Foundation.framework`
- `CommonCrypto.framework`
- `UIKit.framework`

For more information about setting up frameworks as required, visit https://developer.apple.com.

Import PPRiskMagnes.framework

You must add the library to the application for accessing the APIs in the app. The library is contained in an added framework. To import the Swift library, link `PPRiskMagnes.framework` to the app, go to **Build Phases** and select **Link Binaries with Libraries**.

Set up Magnes for Swift

Magnes must be set up at app startup. The `setUp` call is executed only once per lifecycle of a Magnes Singleton Instance. The Setup Parameters below are optional. You can set it up with empty string values. Use the parameters with which you are familiar, otherwise build a default setup without using any parameter.

The `setUp` call either passes the merchant app's `app_guid`, or Magnes creates an `app_guid` to identify the app. An `app_guid` is a unique installation identifier of the app for a particular mobile device. If the app is reinstalled, a new `app_guid` is generated.

Magnes setup Swift App to Swift SDK – simple

```
MagnesSDK magnesSDK = MagnesSDK.shared()
magnesSDK.setup()
```

Magnes setup Swift App to Swift SDK – advanced

```
//@objc public func setUp(setOptionalAppGuid appGuid: String = "",
// disableRemoteConfiguration isDisabled: Bool = false,
// magnesSource source: MagnesSource =
// MagnesSource.DEFAULT)


MagnesSDK magnesSDK = MagnesSDK.shared()
magnesSDK.setUp(setOptionalAppGuid: "YOUR-APPGUID",
setOptionalAPNToken:"YOUR-APP-NOTIFICATION-TOKEN",
disableRemoteConfiguration: false,
magnesSource: MagnesSource)
```

Magnes setup Objective-C App to Swift MagnesSDK

Import the exposed classes from library using the following statement:

```
#import <PPRiskMagnes/PPRiskMagnes-Swift.h>
```

```
//@objc public func setUp(setEnviroment env: Environment = Environment.LIVE,
// setOptionalAppGuid appGuid: String = "",
// setOptionalAPNToken apnToken: String = "",
// disableRemoteConfiguration isRemoteConfigDisabled:
// Bool = false,
// disableBeacon isBeaconDisabled: Bool = false,
// magnesSource source: MagnesSource =
// MagnesSource.DEFAULT)

MagnesSDK *magnessdk = [MagnesSDK shared];
[magnesSDK setUpWithSetEnviroment:EnvironmentLIVE
setOptionalAppGuid:@"YOUR-APPGUID"
setOptionalAPNToken:@"YOUR-APNS-TOKEN"
disableRemoteConfiguration:false
disableBeacon:false
magnesSource:MagnesSourceDEFAULT];
```

Setup parameters and methods

Parameter Name	Data Type	Description

`setAppGuid()`	String	Sets an application's globally unique identifier, which identifies the merchant application that sets up Magnes on the mobile device. If the merchant app does not pass an `AppGuid`, Magnes creates one to identify the app. If the app is installed or reinstalled, it receives a new `AppGuid`. Maximum length: 36 characters.
`SetOptionalAPNToken`	String	Use an Apple Push Notification Service (APNS) token to send notifications to the mobile device
`DisableRemoteConfiguration`	Boolean	Enables the consumer app to receive the latest Magnes configuration settings. The default is `false`.
`MagnesSource`	MagnesSource	Integration defined sources: `MagnesSource.PAYPAL`, `MagnesSource.EBAY`, and `MagnesSource.BRAINTREE`. If none of these apply to your integration, either use `MagnesSource.DEFAULT` or do not set a source.

Collect and submit the payload

When the user launches the mobile app, Magnes remains active only while the `Setup`, `Collect`, and `CollectAndSubmit` methods are called. It does not passively collect data in the background.

You can turn on the debugging log to ensure that Magnes is running successfully.

Magnes generates and returns a new `PayPal-Client-Metadata-Id`, which is a unique 32-character string. Upon collecting the core and dynamic data, Magnes causes the library to submit an asynchronous payload to PayPal Risk Services.

Ordinarily, Magnes generates the `PayPal-Client-Metadata-Id`, but you can pass in a value to be used as the ID.

Collect data

Execute the following code snippet to trigger the library to collect the payload data.

Swift App to Swift SDK: Collect data – simple

```
let magnesResult:MagnesResult = MagnesSDK.shared().collect()
```

Swift App to Swift SDK: Collect data – advanced

```
let magnesResult:MagnesResult = MagnesSDK.shared().collect(cmid: String,
additionalData: [String: String])
```

Objective C App to Swift Library: Collect data – simple

```
MagnesResult *magnesResult = [[MagnesSDK shared] collect];
```

Objective C App to Swift Library: Collect data – advanced

```
MagnesResult *magnesResult = [[MagnesSDK shared]
collectWithPayPalClientMetadataId:"YOUR-TRACKING-ID"
withAdditionalData: (NSDictionary<NSString *,NSString *>)];
```

Collect and submit data

Execute the following code snippets to trigger the library to collect the payload data and send it to the PayPal server.

Swift App to Swift SDK: Collect and submit – simple

```
let magnesResult:MagnesResult = MagnesSDK.shared().collectAndSubmit()
```

Swift App to Swift SDK: Collect and submit – advanced

```
// @objc public func collect(withPayPalClientMetadataId cmid: String,
// withAdditionalData additionalData: [String: String])
// -> MagnesResult

let magnesResult:MagnesResult =
MagnesSDK.shared().collectAndSubmit(withPayPalClientMetadataId:
"YOUR-TRACKING-ID", withAdditionalData: [String: String])
```

From Objective C App to Swift SDK – simple

```
MagnesResult *magnesResult = [[MagnesSDK shared] collectAndSubmit];
```

From Objective C App to Swift SDK – advanced

```
Collect and Submit: Advanced
//@objc public func collectAndSubmit(withPayPalClientMetadataId cmid: String,
// withAdditionalData additionalData: [String: String])
// -> MagnesResult
//

MagnesResult *magnesResult = [[MagnesSDK shared]
collectAndSubmitWithPayPalClientMetadataId:(NSString *)
withAdditionalData:(NSDictionary<NSString *,NSString *>)];
```

Submit additional data

The merchant can include any additional wanted key-value pair data in the Magnes payload. To pass additional data as a dictionary of key-value pairs, execute the above code snippet with the following parameters.

The following table lists parameters you can pass in the payload, including additional wanted data as key-value pairs.

Parameter	Data Type	Description
`Tracking Id`	String	Your own unique identifier for the payload. If you do not pass in this value, a new `Tracking Id` is generated per method call. Maximum length: 32 characters.
`withAdditionalData`	String	Any key-value pair in `NSDictionary` is injected into the payload submitted to the PayPal server

Get the Magnes result

In every data collection call, the Magnes Library returns to the caller a `MagnesResult` containing the latest device information and the `Tracking Id`:

Get the MagnesResult from Swift app to Swift SDK

```
// cached the MagnesResult from previous collectAndSubmit call.

let magnesResult:MagnesResult = MagnesSDK.shared().collectAndSubmit()

let paypalcmid:String = magnesResult.getPayPalClientMetaDataId()
var magnesPayload: [String: Any] = magnesResult.getDeviceInfo()
```

The following table lists methods you can use in a data collection call.

Method	Data Type	Description
`getPaypalClientMetaDataId()`	String	The newly generated (or passed in) `Tracking Id` from the latest API call
`getDeviceInfo()`	`NSDictionary`	A full device information payload. This payload is identical to the payload submitted to the PayPal server.

Get the MagnesResult from from Objective C app to Swift library

```
// cached the MagnesResult from previous collectAndSubmit call.
// MagnesResult *magnesResult = [[MagnesSDK shared] collectAndSubmit];

NSString* cmid = [magnesResult getPayPalClientMetaDataId];
NSDictionary* payloadDict = [magnesResult getDeviceInfo];
```

Method	Data Type	Description
`getPaypalClientMetaDataId()`	String	The newly generated (or passed in) `Tracking Id` from the latest API call
`getDeviceInfo()`	JSONObject	A full device information payload. This payload is identical to the payload submitted to the PayPal server.

Send the PayPal-Client-Metadata-ID from the merchant server to PayPal

The `Tracking Id` pairs the Magnes payload in the context of a PayPal transaction payment, login, or consent, or other PayPal activity.

When the merchant server makes a call to Verifone Transactions APIs, that payment call must include the most recent `Tracking Id` that Magnes (or the merchant app) provided. For most REST APIs, you must include in the call header the `PayPalFraudId` header with the ID's most recent value as that key's value.

iOS Objective-C SDK Integration of Magnes

Before you begin, the following table lists prerequisite information for integrating Magnes on the iOS platform:

Prerequisite	Description
OS version	Using the latest iOS version is recommended

Prerequisite	Description
Software configuration	Magnes iOS Objective-C library supports apps written in Objective-C. PPRMOCMagnesSDK is the central interface for Objective-C library. It provides singleton access to the host app.

Designate frameworks

Before you initialize Magnes, you must set several frameworks as **required** so that Magnes can access essential mobile data for risk assessment. For example, by setting `CoreLocation.framework` as required, Magnes can access the current location of the mobile device.

Set the following frameworks as **required**.

- `Security.framework`
- `CFNetwork.framework`
- `SystemConfiguration.framework`
- `MessageUI.framework`
- `CoreLocation.framework`
- `Foundation.framework`
- `CommonCrypto.framework`
- `UIKit.framework`

For more information about setting up frameworks as required, visit https://developer.apple.com.

Import the library

You must add the library to the application for accessing the APIs in the app. The library is contained in a framework you can add. To import the Objective-C library, link `libPPRiskMagnesOC.a` to the app, then go to **Build Phases** and select **Link Binaries with Libraries**. Import the following files from the Objective-C library: `PPRMOCMagnesSDK.h`, `PPRMOCMagnesSDKResult.h`

Set up Magnes for Objective-C

Magnes must be set up at app startup. The `setUp` call is executed only once per lifecycle of a Magnes Singleton Instance. The Setup Parameters below are optional. You can set it up with empty string values.

The `setUp` call either passes the merchant app's `app_guid`, or Magnes creates an `app_guid` to identify the app. An `app_guid` is a unique installation identifier of the app for a particular mobile device. If the app is reinstalled, a new `app_guid` is generated.

Magnes setup – simple

```
PPRMOCMagnesSDK *PPRMOCMagnesSDK = [PPRMOCMagnesSDK shared];
```

Magnes setup – advanced

```
- (void)setUpOptionalAppGuid:(NSString *)appGuid
withOptionalAPNToken:(NSString *)apnToken
disableRemoteConfiguration:(Boolean)isDisabled
forMagnesSource:(MagnesSourceFlow)magnesSource;

PPRMOCMagnesSDK *PPRMOCMagnesSDK = [PPRMOCMagnesSDK shared];
[_PPRMOCMagnesSDK setOptionalAppGuid: "YOUR-APPGUID",
withOptionalAPNToken:"YOUR-APP-NOTIFICATION-TOKEN",
disableRemoteConfiguration: false,
forMagnesSource: MAGNES_SOURCE_DEFAULT];
```

Setup parameters and methods

Parameter Name	Data Type	Description
`setOptionalAppGuid`	String	Sets an application's globally unique identifier, which identifies the merchant application that sets up Magnes on the mobile device. If the merchant app does not pass an `AppGuid`, Magnes creates one to identify the app. If the app is installed or reinstalled, it receives a new `AppGuid`. Maximum length: 36 characters.
`withOptionalAPNToken`	String	Use an Apple Push Notification Service (APNS) token to send notifications to the mobile device.
`disableRemoteConfiguration`	Boolean	Enables the consumer app to receive the latest Magnes configuration settings. The default is `false`.
`forMagnesSource`	MagnesSource	Integration defined sources: `MAGNES_SOURCE_PAYPAL` = `10`, `MAGNES_SOURCE_EBAY` = `11`, `MAGNES_SOURCE_BRAINTREE` = `12`, `MAGNES_SOURCE_DEFAULT` = `-1`

Collect and submit the payload

When the user launches the mobile app, Magnes remains active only while the `Setup`, `Collect`, and `CollectAndSubmit` methods are called. It does not passively collect data in the background.

You can turn on the debugging log to ensure that Magnes is running successfully.

Magnes generates and returns a new `Tracking Id`, which is a unique 32-character string. Upon collecting the core and dynamic data, Magnes triggers the library, which transmits an asynchronous payload to PayPal Risk Services.

Ordinarily, Magnes generates the `Tracking Id`, but you can pass in a value to be used as the ID.

Collect data from Objective-C app

Execute the following code snippets to trigger the Objective-C library to collect the payload data.

Collect data – simple

```
//PPRMOCMagnesSDK *magnesSDK = [PPRMOCMagnesSDK shared];
PPRMOCMagnesSDKResult *magnesResult = [magnesSDK collect];
```

Collect data – advanced

```
//PPRMOCMagnesSDK *magnesSDK = [PPRMOCMagnesSDK shared];
PPRMOCMagnesSDKResult *magnesResult =
[magnesSDK collectWithPayPalClientMetadataId:"YOUR-TRACKING-ID"
withAdditionalData:@{}];
```

Collect and submit data from Objective-C app

Execute the following code snippet to trigger the library to collect the payload data and send it to the PayPal server.

Collect and submit – simple

```
//PPRMOCMagnesSDK *magnesSDK = [PPRMOCMagnesSDK shared];
PPRMOCMagnesSDKResult *magnesResult = [magnesSDK collectAndSubmit];
```

Collect and submit – advanced

```
//PPRMOCMagnesSDK *magnesSDK = [PPRMOCMagnesSDK shared];
PPRMOCMagnesSDKResult *magnesResult =
[magnesSDK
collectAndSubmitWithPayPalClientMetadataId:@"YOUR-TRACKING-ID"
withAdditionalData:@{}];
```

Specifying additional data

The merchant can include any additional wanted key-value pair data in the Magnes payload. To pass additional data as a dictionary of key-value pairs, execute the following code snippets.

Parameter	Data Type	Description
`withPaypalClientMetaDataId id`	String	Your own unique identifier for the payload. If you do not pass in this value, a new `Tracking Id` is generated per method call. Maximum length: 32 characters.
`withAdditionalData data`	[String: String]	Any key-value pair in this dictionary is injected into the payload submitted to the PayPal server

Get the Magnes result

In every data collection call, the Magnes Library returns to the caller a `MagnesResult` containing the latest device information and the `Tracking Id`:

Get the MagnesResult from Objective-C app

```
// cached the MagnesResult from previous collectAndSubmit call.
PPRMOCMagnesSDK *magnesSDK = [PPRMOCMagnesSDK shared];
PPRMOCMagnesSDKResult *magnesResult = [magnesSDK collectAndSubmit];

NSString *payPalClinetMetaDataId = [magnesResult getPayPalClientMetaDataId];
NSDictionary *deviceInfo = [magnesResult getDeviceInfo];
```

The following table lists methods you can use in a data collection call.

Method	Data Type	Description
`getPaypalClientMetaDataId()`	String	The newly generated (or passed in) `Tracking Id` from the latest API call
`getDeviceInfo()`	JSONObject	A full device information payload. This payload is identical to the payload submitted to the PayPal server.

Send the PayPal-Client-Metadata-ID from the merchant server to PayPal

The `Tracking Id` pairs the Magnes payload in the context of a PayPal transaction payment, login, or consent, or other PayPal activity.

When the merchant server makes a call to Verifone Transactions APIs, that payment call must include the most recent `Tracking Id` that Magnes (or the merchant app) provided. For most REST APIs, you must include in the call header the `PayPalFraudId` header with the ID's most recent value as that key's value.

For other APIs, refer to the API documentation or integration details provided by your PayPal representative.

Data collection, usage, and privacy

Magnes collects mobile device data based on the permissions granted during the installation of the mobile app.

Data collected by Magnes is used for risk analysis and authentication. Verifone and PayPal do **not** share Magnes data with third parties for their own benefit.

Sample payloads

The following are sample payloads as an aid in interpreting the results of your Magnes app. The samples include one payload each of Android and iOS data, listed by parameter names in alphabetical order.

Android sample payload


```
{
"android_id": "e54d2783a24c5a93",
"app_first_install_time": 1485391650466,
"app_guid": "16dd2f5f-3496-4953-8021-00b45950bdeb",
"app_id": "com.paypal.android.lib.riskcomponentsample",
"app_last_update_time": 1486085153926,
"app_version": "1.0",
"bssid": "6c:f3:7f:b9:1e:90",
"bssid_array": ["6c:f3:7f:b9:1e:90","6c:f3:7f:b9:1e:92"],
"cdma_network_id": -1,
"cdma_system_id": -1,
"cell_id": 21181955,
"comp_version": "3.5.7.release",
"conf_url": "https://www.paypalobjects.com/webstatic/risk/dyson_config_android_v3.json",
"conf_version": "3.0",
"conn_type": "MOBILE",
"dc_id": "50fd71cdaeb32cfc7421703646d30489",
"device_id": "352530080310824",
"device_model": "Pixel XL",
"device_name": "marlin",
"device_uptime": 798843490,
"ds": false,
"gsf_id": "3d6eefb279e2e84a",
"ip_addresses": ["192.0.0.4","fe80::c8ec:37ff:fe6b:6141%dummy0","2607:fb90:a74b:2d64:ba05:3877:ad93:4bdd",
"fe80::ba05:3877:ad93:4bdd%rmnet_data0","fe80::5df9:65ee:ffa6:96ce %rmnet_data7",
"fe80::a918:e8bd:ad79:d8cc%r_rmnet_data0"],
"ip_addrs": "192.0.0.4",
"is_emulator": false,
"is_rooted": false,
"known_apps": ["com.my-org.mobile/com.my-org.mobile.activities. my-org"],
"linker_id": "f76d48e7-63f8-4be9-bd4d-08edaa4d59b3",
"locale_country": "US",
"locale_lang": "en",
"location": {"lat":37.3760446,"lng":-121.9216664,"acc":19.872,"timestamp":1486085186851},
"location_area_code": "14940",
"location_auth_status": "unknown",
"mac_addrs": "02:00:00:00:00:00",
"network_operator": "310260",
"notif_token": "test notif token",
"os_type": "Android",
"os_version": "7.1.1",
"pairing_id": "1d7706b1f3ba46bdbe46832aa99852cf",
"payload_type": "full",
```

```
"phone_type": "gsm",
"pm": "fe3e",
"risk_comp_session_id": "1fe8a38f-c4ad-4cbe-84bb-e340d953160b",
"roaming": false,
"serial_number": "HT68J0207231",
"sim_operator_name": "T-Mobile",
"sim_serial_number": "8901260533554717229",
"sms_enabled": true,
"source_app": 0,
"source_app_version": "1.9.9",
"ssid": "PayPalGuest",
"subscriber_id": "310260535471755",
"timestamp": 1486085312082,
"total_storage_space": 26109874176,
"tz": "-28800000",
"tz_name": "Pacific Standard Time",
"vpn_setting": "tun0"
}
```


iOS sample payload


```
{
"app_guid": "b286802c-de9d-4c30-8ecb-42c2d2dfe3e4",
"app_id": "com.paypal.Dyson",
"app_version": "1.0",
"c": "48",
"cloud_identifier": "7027fff5-fde0-44ab-8b25-6cb4a928de8f",
"comp_version": "3.5.7",
"conf_url": "https:\/\/www.paypalobjects.com\/webstatic\/risk\/dyson_config_ios_v4.json",
"conf_version": "4.0",
"conn_type": "wifi",
"dc_id": "11e7d85632c3142b07ffeee4516629a3",
"device_model": "Simulator",
"device_name": "Eddie's Papa",
"ds": false,
"email_configured": false,
"ip_addresses": ["::1","127.0.0.1","fe80::1","fe80::c6b3:1ff:febd:30c9","10.225.90.250",
"fe80::c034:cdff:fe5c:13e3"],
"ip_addrs": "10.225.90.250",
"is_emulator": true,
"is_rooted": false,
"known_apps": ["com.my-org.mobile/com.my-org.mobile.activities.my-org"],
"linker_id": "a0cefbe4-b445-47d1-a964-bd39e8c0ea8d",
"local_identifier": "0579a19c-f302-4242-a481-a2f49689a6ff",
"locale_country": "US",
"locale_lang": "en",
"location": {"lng":-122.03076342,"lat":37.33123666,"timestamp":1486084281488,"acc":30},
"location_auth_status": "denied",
"notif_token": "null",
"os_type": "iOS",
"os_version": "10.1",
"pairing_id": "b2d32ea3df86477da25523d1cc19d37b",
"payload_type": "full",
"pin_lock_last_timestamp": null,
"pm": "338723cb",
"risk_comp_session_id": "8fb8114e-2079-436a-bec9-583bffcb108c",
"sms_enabled": false,
"source_app": 10,
"source_app_version": "1.0",
"timestamp": 1486084490678,
```

```
"total_storage_space": 499082485760,
"tz": "-28800000",
"tz_name": "America\/Los_Angeles",
"vendor_identifier": "20C722AF-BDC7-4107-8C07-5FA1875AA7F1",
"vpn_setting": "tun0"
}
```