

# **Extended OBU Library (SDK) Developer Guide**

**Third Party Application for Android & iOS**

Revision 1.3  
29 September 2023

Copyright © 2023 by Land Transport Authority. All rights reserved.

All trademarks or registered trademarks mentioned in this document are properties of their respective owners.

No part of this document may be reproduced or transmitted in any form or by any means, electronic, mechanical or otherwise, without a prior written permission.

If you are not the intended recipient, you are hereby notified that any disclosure, copying, distribution, or the taking of any action based on the contents of this information is strictly prohibited. If you received this document by error, you are kindly requested to immediately contact us.

## Revision History

Revision	Effective Date	Description
1.0	01 Mar 2023	First Release
1.1	01 May 2023	Minor Revision
1.2	30 Jun 2023	Updated Sections: <ul style="list-style-type: none"> <li>- Updated Section 1.3 Description</li> <li>- Updated Section 3.8.3.1 Code Snippet and Description</li> <li>- Updated Section 3.8.3.5 Function List Table</li> <li>- Updated Section 3.8.4 to add Function for item 10</li> <li>- Added Section 3.8.6 Traffic Icons in SDK for Android</li> <li>- Updated Section 4.7 Code Snippet</li> <li>- Updated Section 4.7.6 Function List Table</li> <li>- Added Section 4.7.8 Traffic Icons in SDK for iOS</li> <li>- Updated Section 5.4 to remove content4 from Function List Table</li> <li>- Updated Section 5.7 Parameter Names</li> <li>- Moved Section 6.8.11 OBUDData to Section 6.8</li> <li>- Moved Section 6.8 Enums to Section 6.9</li> <li>- Added Appendix A for TDCID icons</li> </ul>
1.3	29 Sep 2023	Updated “nexgen ERP” to “ERP 2.0” Updated Sections: <ul style="list-style-type: none"> <li>- Update Section 3.5 Code Snippet</li> <li>- Update Section 3.8.3.1 to note of using location services</li> <li>- Update Section 3.8.3.5, item 5 - Travel Summary description</li> <li>- Update Section 4.7.5.2, item 7 - Travel summary description</li> <li>- Update Section 4.7.5.2, added note on item 8 – TotalTripCharged</li> </ul>

## Table of Contents

1. Overview .....	8
1.1 Electronic Road Pricing .....	8
1.2 Point-Based Charging .....	9
1.3 Electronic Parking System .....	10
1.4 Enhanced Electronic Parking .....	11
1.5 Common Alert .....	11
1.6 Traffic Messages .....	12
2. Getting Started .....	14
2.1 Getting SDK Account Key/Sign Up .....	14
2.2 Sample Application with SDK Integration .....	14
2.3 Register Your Phone at e-Services Portal .....	15
2.3.1 MAC Address Authentication for Android Phones .....	15
2.3.2 MAC Address Authentication for iPhones .....	15
3. Android .....	16
3.1 Minimum API Level for the SDK .....	16
3.2 Programming Language .....	16
3.3 Bluetooth Requirement .....	16
3.4 Third-Party Libraries .....	16
3.5 How to Add the SDK to Android Application .....	17
3.6 SDK Permissions .....	18
3.6.1 SDK Custom Permission for Third-Party Applications .....	18
3.7 Other Requirements .....	19
3.8 How to Integrate the SDK .....	19
3.8.1 Step 1: Initialise the SDK with an SDK Account Key .....	19
3.8.2 Step 2: OBU Data Permission .....	21
3.8.3 Step 3: Establish a Connection to the OBU .....	21
3.8.4 SDK Helper Methods for OBU Data .....	30
3.8.5 Mock Manager for Android Applications .....	31
3.8.6 Traffic Icons in the OBU SDK .....	33
4. iOS .....	34
4.1 Minimum iOS Version .....	34
4.2 Programming Language .....	34
4.3 Bluetooth Requirement .....	34

4.4	Third-Party Libraries.....	34
4.5	How to Add the SDK to iOS Applications .....	34
4.6	SDK Permissions .....	35
4.7	How to Integrate the SDK .....	35
4.7.1	Step 1: Initialise SDK with the SDK Account Key.....	36
4.7.2	Step 2: OBU Data Permission .....	36
4.7.3	Step 3: Search for OBUs.....	37
4.7.4	Connect to an OBU .....	38
4.7.5	Connection Status Delegate.....	38
4.7.6	SDK Helper Methods for OBU Data .....	42
4.7.7	Mock Manager for iOS Applications .....	43
4.7.8	Traffic Icons in SDK for iOS.....	45
5.	Mock Manager for Testing Applications .....	46
5.1	Electronic Road Pricing (ERP) .....	46
5.1.1	Point-Based Charging.....	47
5.1.2	ERP1-Based Charging .....	47
5.1.3	Common Alert.....	47
5.1.4	Electronic Enhanced Parking (EEP).....	48
5.1.5	Electronic Parking System (EPS).....	48
5.1.6	Traffic Messages.....	49
6.	API Classes.....	57
6.1	OBU .....	57
6.2	OBUAcceleration.....	57
6.3	OBUChargingInformation .....	58
6.4	OBUPaymentHistory .....	59
6.5	OBUTravelSummary .....	59
6.6	OBUTotalTripCharged .....	60
6.7	Traffic Information .....	61
6.7.1	Usage in Android .....	61
6.7.2	Usage in iOS .....	62
6.8	OBUData.....	64
6.9	Enums .....	65
6.9.1	OBUCardStatus.....	65
6.9.2	OBUBusinessFunction.....	65

6.9.3	OBUSChargingMessageType .....	66
6.9.4	OBUSChargingPaymentMode.....	66
6.9.5	OBUSChargingType .....	66
6.9.6	OBUSPaymentMode.....	67
6.9.7	OBUSTextColor.....	67
6.9.8	OBUSTextStyle .....	68
6.9.9	OBUSTrafficDataPriority .....	68
6.9.10	OBUSTrafficMessageType .....	69
6.9.11	BluetoothState (iOS).....	70
6.9.12	BluetoothState (Android) .....	70
6.9.13	OBUSErrorCode.....	71
7.	SDK Troubleshooting .....	73
Appendix A:	TDCID Icons .....	74

## Table of Figures

Figure 1-1	EXTOL SDK Flow Overview.....	8
Figure 1-2	Point Based Charging.....	9
Figure 1-3	Electronic Parking System.....	10
Figure 1-4	Enhanced Electronic Parking.....	11
Figure 1-5	Common Alert Display .....	11
Figure 1-6	General Traffic Information.....	12
Figure 1-7	Travel Time Information .....	12
Figure 1-8	Parking Lot Information .....	13
Figure 1-9	Developer Workflow Overview.....	14
Figure 1-10	MAC Address Input Screen for iOS .....	15
Figure 1-11	OBUS Search Screen .....	23

## Abbreviations

Abbreviation	Definition
APK	Android Package
API	Application Programming Interface
AZ	Alert Zone
CDZ	Charging Display Zone
CPO	Car Parking Owner
ERP 2.0	Electronic Road Pricing 2.0 system
ERP1	Current Electronic Road Pricing system with gantries
SDK	Extended OBU Library (EXTOL) Software Development Kit
IU	In-vehicle Unit
OBU	On Board Unit

## 1. OVERVIEW

The Extended OBU Library (EXTOL SDK) aims to enhance your applications by providing APIs to connect and receive ERP (Electronic Road Pricing) and Traffic messages from the OBU while driving on the roads to help improve the driving experience. It is intended for third-party developers and companies who want to integrate OBU Display messages into their applications, enabling them to serve as an alternative display for motorists. The SDK provides high level APIs that make integration into your application swift and straightforward. Communication between the SDK and OBU device is accomplished via Bluetooth.

This guide covers all the details required for integration into your applications for Android and iOS platforms. The diagram below explains the high-level communication between the devices:



Figure 1-1 EXTOL SDK Flow Overview



Note: OBU can only connect to one SDK at a time.

### 1.1 Electronic Road Pricing

In the Electronic Road Pricing 2.0, the roads are divided into virtual segments such as:

- i. **AZ (Alert Zone):** This informs the motorist that there will be charges incurred ahead of the zone.
- ii. **CDZ (Charging Display Zone):** This provides the charging zone information. The OBU sends two back-to-back messages at this point:
  - a. **Charging Information:** the amount that will be deducted at the CDZ.
  - b. **Deduction Success/Failure:** the actual deducted amount at the CDZ for successful and unsuccessful deduction.



## 1.2 Point-Based Charging

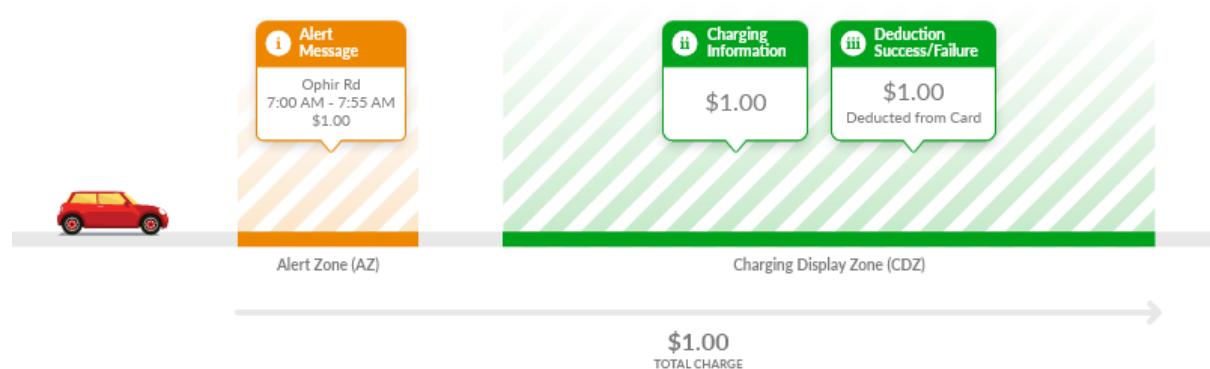


Figure 1-2 Point Based Charging

In Point-Based charging, charging takes place at a certain point on the road and the OBU sends the following messages:

- i. **Point-Based Alert:** This message informs the motorist on the charges that will be incurred during a specific timeframe at the CDZ. It displays the road name, time range and road pricing charge.
- ii. **Point-Based Charging Information:** This message displays the amount that will be deducted at the CDZ.
- iii. **Point-Based Deduction Successful/Failure:** This message displays the actual deducted amount at the CDZ. The message contains information regarding successful and unsuccessful deductions.

### 1.3 Electronic Parking System

Electronic Parking System enables vehicles with In-vehicle Unit (IU) to gain entry and exit car parks. Parking charges are charged based on season parking pass or thru CashCard/CEPAS Card that is inserted in vehicles' IU. In ERP 2.0, the current In-vehicle Units (IU) is replaced with OBU which come with screen that display the charging information and deduction if vehicles do not have the season parking pass.

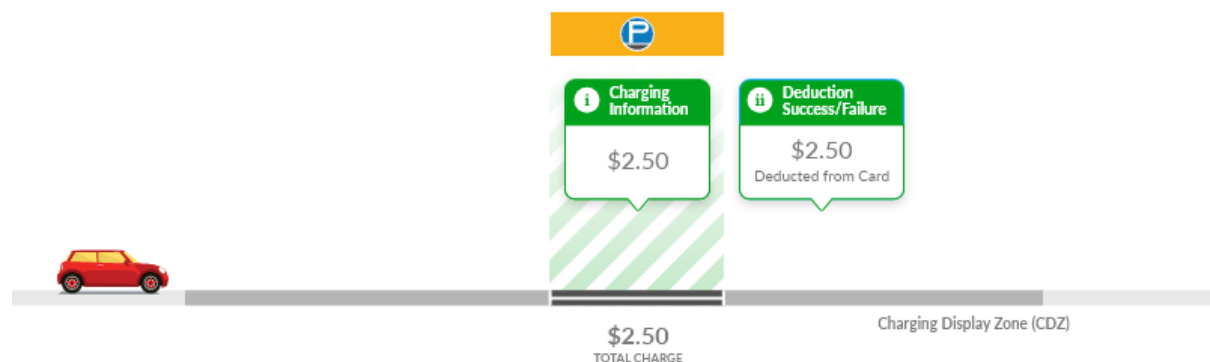


Figure 1-3 Electronic Parking System

- i. **Entry-Exit Charging Information:** This informs the motorist of the amount that will be deducted upon entering the parking area.
- ii. **Entry-Exit Deduction Successful/Failure:** This message displays the actual deducted amount at the exit point of a parking area when successful, and the expected deduction amount when unsuccessful.

## 1.4 Enhanced Electronic Parking

Enhanced Electronic Parking System is the enhancement of Electronic Parking System. Unlike in EPS system, EEP has CPO (Car Parking Owner) information and charging information.

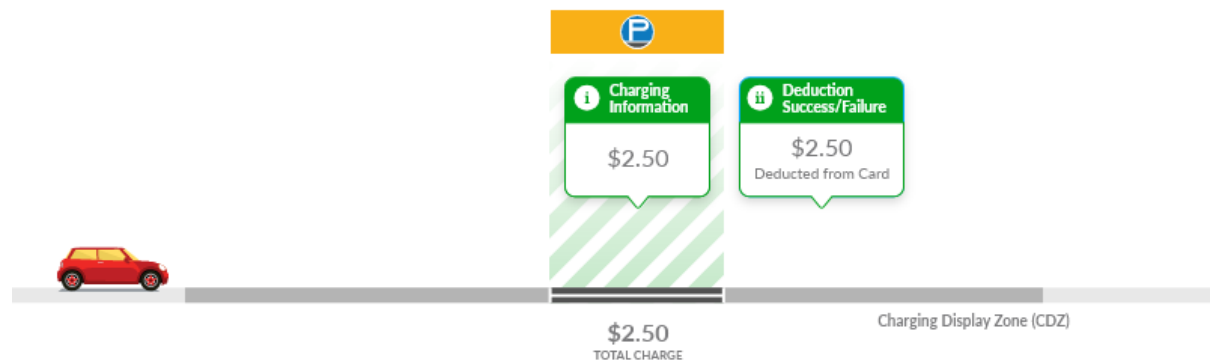


Figure 1-4 Enhanced Electronic Parking

- i. **Entry-Exit Charging Information:** This informs the motorist of the amount that will be deducted upon entering the parking area.
- ii. **Entry-Exit Deduction Successful/Failure:** This message displays the actual deducted amount at the exit point of a parking area when successful, and the expected deduction amount when unsuccessful.

## 1.5 Common Alert

This will display the common alerts available for the ERP 2.0. The message is used to send out road information, as shown in the image.

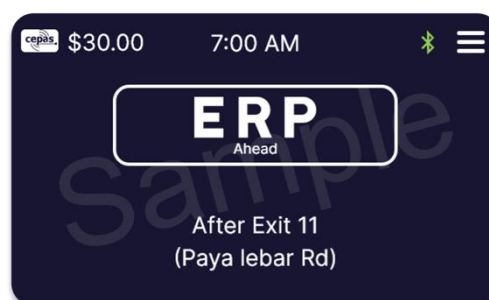


Figure 1-5 Common Alert Display

## 1.6 Traffic Messages

The SDK provides a set of Traffic related messages that are sent out from the OBU while on the road. These messages contain information about traffic conditions ahead of motorists and are divided into three categories:

### i. General Traffic Information



Figure 1-6 General Traffic Information

This group contains messages related to traffic safety information, traffic jams ahead, speed limit information, school zones, silver zones and emergency messages. Typically, the data sent by the OBU includes an image that depicts the type of message, as well as a few rows of text that provide more information about the event.

### ii. Travel Time Information

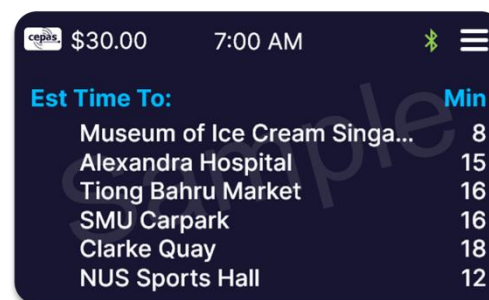
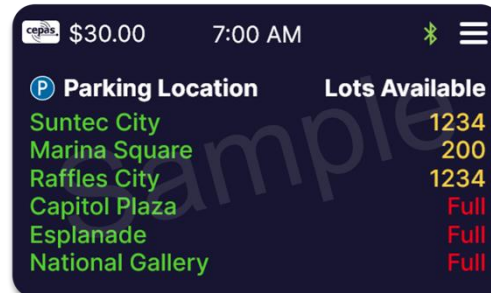


Figure 1-7 Travel Time Information

This message is sent to the motorist and provides an estimate of the travel time to popular locations based on their current location and heading.

### iii. Parking Lot Information



Parking Location	Lots Available
Suntec City	1234
Marina Square	200
Raffles City	1234
Capitol Plaza	Full
Esplanade	Full
National Gallery	Full

Figure 1-8 Parking Lot Information

This message informs the motorist of available parking lots near the motorist's current location.

## 2. GETTING STARTED

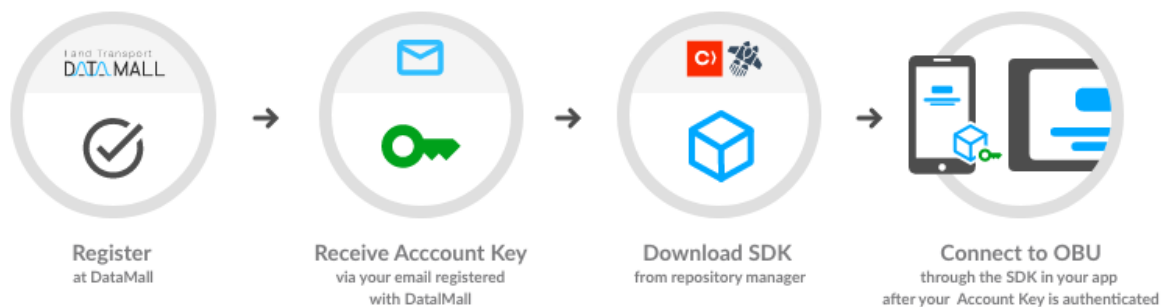


Figure 1-9 Developer Workflow Overview

These are the steps required for developer to get started with the SDK:

- i. Register at DataMall (<https://datamall.lta.gov.sg/content/datamall/en/request-for-api.html>).
- ii. After registering, the developer will receive an email containing the SDK account key and a link to the developer guide.
- iii. Follow the guide to integrate the SDK into your applications.

### 2.1 Getting SDK Account Key/Sign Up

To get started with the SDK, you will need an SDK Account key which can be obtained by registering at DataMall. After submitting the form, you will receive two emails:

- i. The first email will contain the DataMall API Access Key, which you need to access the DataMall APIs.
- ii. The second email will contain the SDK Account Key for the SDK.

### 2.2 Sample Application with SDK Integration

The sample application demonstrates the integration of the SDK into iOS and Android applications. It will be available on github at a later stage.

## 2.3 Register Your Phone at e-Services Portal

The motorist needs to register their iPhone or Android phone's MAC Address at e-Services portal. You can register up to a maximum of five devices per OBU. Once registered, the OBU will authenticate the MAC addresses of the paired phones against the registered list. Further details on MAC address authentication for both iOS and Android can be found below.

### 2.3.1 MAC Address Authentication for Android Phones

During Bluetooth pairing, Android OS automatically shares the phone's MAC address with the OBU. The OBU will then verify the MAC address against the list of registered addresses on the e-Services Portal for authentication. If authentication fails, an error will be displayed.

### 2.3.2 MAC Address Authentication for iPhones

When pairing an iPhone with the OBU, the device's MAC address must be manually inputted by the motorist through an on-screen prompt (Figure 1-10). The OBU will then verify the MAC address against the list of registered addresses on the e-Services Portal for authentication.

Please ensure that you enter the MAC address registered to the OBU on the screen below. The SDK will display this prompt whenever you connect to a new OBU.

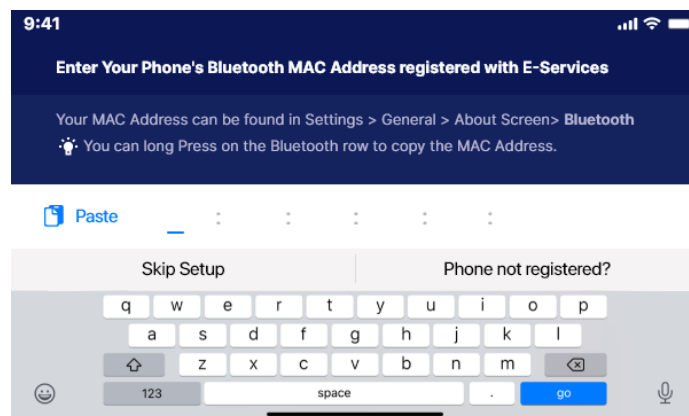


Figure 1-10 MAC Address Input Screen for iOS

## 3. ANDROID

### 3.1 Minimum API Level for the SDK

The SDK requires minimum API Level 26.

```
android {  
    defaultConfig {  
        minSdkVersion 26  
        ...  
    }  
}
```

### 3.2 Programming Language

The SDK has been developed in Kotlin. The recommended language of third-party applications is Kotlin. However, it could be used in Java applications.

### 3.3 Bluetooth Requirement

The communication between the OBU device and the SDK happens via Bluetooth. The OBU supports Bluetooth 1.0, 2.0+EDR, 4.0.

### 3.4 Third-Party Libraries

The SDK uses the following third-party libraries:

- i. Retrofit (v2.9.0)
- ii. SQLCipher (v4.4.3)
- iii. GSON (v2.9.0)



### 3.5 How to Add the SDK to Android Application

To include the SDK in your Android application, you can add it as a Maven dependency. Add the following to your top-level build.gradle file.

```
repositories {  
    google()  
    jcenter()  
    maven {  
        url 'https://extol.mycloudrepo.io/public/repositories/extol-android'  
    }  
}
```

Then, in your application's build.gradle file, add the following dependency:

```
dependencies {  
    ...  
    implementation 'sg.gov.lta:extol:1.1.0 '  
    ...  
}
```

## 3.6 SDK Permissions

The SDK requires the following permissions:

- i. Internet
- ii. Bluetooth permissions
- iii. OBU Data Access permission (OBU SDK custom permission)

### 3.6.1 SDK Custom Permission for Third-Party Applications

To maintain security of collecting data from the OBU for the motorist, the SDK needs explicit consent from the motorist in the form of a permission. The SDK requires applications to take the following custom permission from motorists:

```
<app's package name>.permission.OBU_DATA_ACCESS
```

For example, if your application's package is **com.example.myapp**, then the custom permission would be **com.example.myapp.permission.OBU\_DATA\_ACCESS**. This permission needs to be requested at runtime by the developer of an application using the SDK. Without granting this permission, the SDK methods will throw a run time exception.

The SDK requires the following permissions, which will be merged into your application's manifest after integrating the SDK. Therefore, you do not need to manually add them to your application's manifest:

```
<uses-permission android:name="android.permission.BLUETOOTH" />  
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />  
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />  
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

### 3.7 Other Requirements

The SDK utilises the Companion Device API to scan for nearby OBUs. This API indicates to the system that your application intends to establish companion device connections.

```
<uses-feature
    android:name="android.software.companion_device_setup"
    android:required="true" />
```

### 3.8 How to Integrate the SDK

#### 3.8.1 Step 1: Initialise the SDK with an SDK Account Key

The first step is to initialise the SDK with your SDK Account Key, which is sent to you after registration at DataMall. The function takes an optional OBUInitListener to give you success and failure callbacks. The SDK will only work after it has been successfully initialised. The `onError()` method returns a list of error codes defined below:

```
val initListener = object : OBUInitListener {

    override fun onSuccess() {
        // Initialization was successful
    }

    override fun onError(error: OBUError) {
        // Initialization failed with error
    }
}

OBUSDK.init(context, "<SDK_ACCOUNT_KEY>", initListener)
```

## OBUInitListener:

1	<p><b>fun onSuccess()</b></p> <p>This is invoked upon the successful authentication of the SDK Account key.</p>
2	<p><b>fun onError(error: OBUError)</b></p> <p>The possible errors are defined below:</p> <ul style="list-style-type: none"> <li>i. <b>INVALID_SDK_ACCOUNT_KEY:</b> if the SDK Account is not valid.</li> <li>ii. <b>INTERNET_NOT_AVAILABLE:</b> There is no internet connectivity on your phone.</li> <li>iii. <b>DEVELOPER_UNAUTHORISED:</b> The developer is not authorised to use the SDK. Please contact support if your SDK Account key has been blocked.</li> <li>iv. <b>APPLICATION_UNAUTHORISED:</b> your application with the SDK Account Key is not authorised. Please contact support if your application has been blocked.</li> <li>v. <b>DEVELOPER_DEACTIVATED:</b> Please contact support if you come across with error.</li> <li>vi. <b>SERVER_ERROR:</b> Server is not reachable. Please try again after some time. If the error persists, please contact support.</li> <li>vii. <b>REQUEST_TIMEOUT:</b> This happens if internet is slow, or server takes longer to reply.</li> </ul>



Note: This method must be called before calling any other SDK APIs, otherwise a runtime exception will be thrown. It is recommended to call `OBUSDK.init()` from your Application class.

### 3.8.2 Step 2: OBU Data Permission

Calling the following method shows the OBU Data permission alert to the motorist. The sample code is shown below.

```
requestPermissions(arrayOf("<app's package name>.permission.OBU_DATA_ACCESS"),  
REQUEST_CODE)  
  
override fun onRequestPermissionsResult(  
    requestCode: Int,  
    permissions: Array<out String>,  
    grantResults: IntArray  
) {  
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)  
    if(requestCode == REQUEST_CODE && grantResults[0] ==  
    PackageManager.PERMISSION_GRANTED) {  
        // startListening() or connect() can be called here  
    }  
}
```

### 3.8.3 Step 3: Establish a Connection to the OBU

To make a connection with the OBU, the first step is to search for OBUs.

#### 3.8.3.1 Search for OBUs

Calling this function starts a search and scan for nearby OBUs. The SDK uses the companion device pairing API to accomplish this task. Once the SDK discovers OBUs, it presents a system pop-up containing a list of OBUs for the motorist to choose from.

The method takes an OBUEventListener to receive callbacks regarding the search.



Note: Location services are required for searching OBUs

```
OBUSDK.startSearch(listener: OBUEventListener)
```

```
val searchListener = object : OBUSearchListener {
    override fun onObuSelected(obu: OBU) {

    }
    override fun onPairing() {

    }

    override fun onSearchFailure(error: OBUError?) {

    }

    OBUSDK.startSearch(searchListener)
```

1	<b>fun onObuSelected(obu: OBU)</b>
	<p>obu: OBU</p> <p>After the motorist selects an OBU from the search dialogue, this method returns an OBU object. The developer can then call the connect() method directly to establish a connection with the selected OBU.</p>
2	<b>fun onPairing()</b>
	<p>This function is invoked when the OBU being connected to has not been previously bonded with the device, indicating that the pairing process with commence.</p>
3	<b>fun onSearchFailure(error: OBUError?)</b>
	<p>This function is called.</p> <ul style="list-style-type: none"> <li>i. <b>OBUIInternalSearchError:</b> when an error occurs in the Companion Device API search. This can happen if devices are not found or if there is a search timeout. If such an error occurs, you are required to restart the search. Please ensure that Bluetooth is enabled on your phone before initiating the search again.</li> <li>ii. <b>OBUBluetoothdisabledError:</b> When Bluetooth is disabled.</li> </ul>

The system pop-up looks like the one shown below. The title on the dialogue '**Link with <APP\_NAME>**' will be replaced by your application name. The list shows the available OBUs that have been discovered during the search. The user can select one of the OBUs to pair with the application.

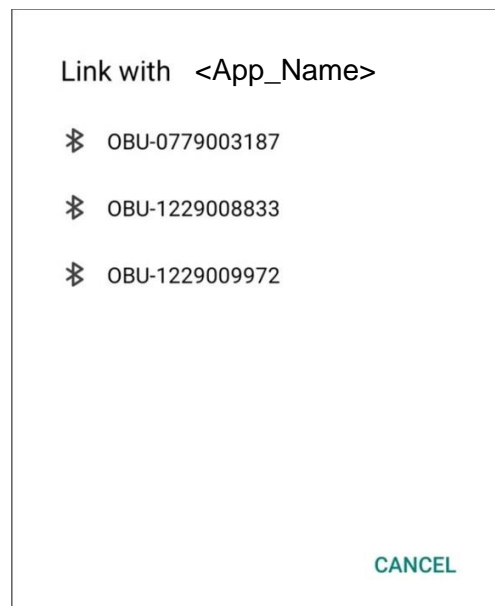


Figure 1-11 OBU Search Screen



Note: The OBU name format is in OBU-XXXXXXXXX

### 3.8.3.2 Connect to the OBU

Once the OBUs are found after calling the search method mentioned above, the next step is to make a connection to one of the OBUs shown in the above dialogue. The Search and connection callbacks are sent in the `OBUConnectionStatusListener` explained in the section below.

The SDK provides two `connect()` methods to establish a connection with the OBU: the first takes an OBU object which can be received from one of the callbacks explained in the `OBUConnectionStatusListener`, and the other takes in a vehicle number of a car registered with the OBU. The details of both methods are explained below.

```
fun connect(obu: OBU, listener: OBUConnectionStatusListener)
```

There is another method for making the connection, which uses the vehicle number to establish the connection. However, this method requires the OBU to be previously paired with the device.

```
fun connect(vehicleNumber: String, listener: OBUConnectionStatusListener)
```

### 3.8.3.3 OBUConnectionStatusListener

The connection listeners provide callbacks related to the connection. All methods in this listener are optional.

```
val connectionListener = object : OBUConnectionStatusListener {

    override fun onOBUConnected(device: OBU) {

    }

    override fun onOBUConnectionFailure(error: OBUError) {

    }

    override fun onOBUDisconnected(device: OBU?, error: OBUError?) {

    }

}

OBUSDK.connect(obu, connectionListener)
```

The description of each method is shown below:

1	<b>fun onOBUConnected(device: OBU)</b>
	This function is called when the connection is established with the OBU.
2	<b>fun onOBUConnectionFailure(error: OBUError)</b>
	error: This function is called when the SDK is unable to establish a connection with the OBU due to one of the possible error codes mentioned below:
	i. <b>OBUIInternalError:</b> Please contact customer support if this error persists.
	ii. <b>OBUNotPairedError:</b> When connecting using vehicle number via <b>fun connect(vehicleNumber, connectionlistener)</b> defined in section 3.8.3.2, this error is thrown if the OBU registered with the Vehicle number has not been paired with the phone.
	iii. <b>OBUSDKDataAccessPermissionDenied:</b> In case if the permission is not granted or revoked. The application must explicitly ask the OBU_DATA_ACCESS permission from the user and the user must “Allow” it.
	iv. <b>OBUIInternetNotAvailableError:</b> Internet is required for some cases. This is returned if internet is not available on the phone (whether WiFi or mobile data).
	v. <b>OBUConnectionFailedError:</b> This is thrown if an error occurs during establishing a Bluetooth connection with an OBU device.
	vi. <b>OBUSDKPublicKeyForOBUNotAvailable(NO_PUBLIC_KEY_AVAILABLE):</b> This is an internal error. If this occurs, make sure that your phone is registered with e-Services portal for your OBU,



3	<b>fun</b> onOBUDisconnected(device: OBU?, error: OBUError?)
	<p>This function is called when the established connection with the OBU has been disconnected due to the possible errors that can be returned are as follows:</p> <ul style="list-style-type: none"> <li>i. <b>OBUIInternalError(SERVER_ERROR)</b>: Some unexpected values/result was returned.</li> <li>ii. <b>OBUSDKPublicKeyForOBUNotAvailable(NO_PUBLIC_KEY_AVAILABLE)</b>: This is an internal error. If this occurs, make sure that your phone is registered with e-Services portal for your OBU,</li> <li>iii. <b>OBUSDKCodeSignatureFail(CODE_SIGNATURE_FAIL)</b>: The data signature validation failed.</li> <li>iv. <b>OBUSDKDataAccessPermissionDenied(OBU_DATA_ACCESS_PERMISSION_DENIED)</b>: This error is returned if the user denies or revokes the SDK Data Access Permissions.</li> <li>v. <b>OBUBadRequestError(BAD_REQUEST)</b>: Something went wrong with the server. Please contact support and try again after some time.</li> <li>vi. <b>OBUIInternetNotAvailableError(INTERNET_NOT_AVAILABLE)</b>: Internet is not available on the device.</li> </ul>

### 3.8.3.4 Bluetooth Listener

This interface provides the change of states of the Bluetooth adapter.

```
private val bluetoothListener = object : OBUBluetoothListener {
    override fun onBluetoothStateUpdated(state: BluetoothState) {
        if (state == BluetoothState.STATE_ON) {

        }
    }
}
```

OBUSDK.setBluetoothListener(bluetoothListener)

1	<b>fun</b> onBluetoothStateUpdated(state: BluetoothState)
	<p>These are the possible states:</p> <p>STATE_TURNING_OFF, STATE_TURNING_ON, STATE_OFF, STATE_ON</p>

### 3.8.3.5 Subscribe to the OBUDataListener

Once the connection has been established, the OBUDataListener is ready to receive data from the OBU. All the methods in the interface are optional. You can implement the data that you want to use in your application from the optional methods in the interface.

```
val dataListener = object : OBUDataListener {

    override fun onChargingInformation(chargingInfo: List<OBUChargingInformation>) {

    }

    override fun onVelocityInformation(velocity: Double) {

    }

    override fun onAccelerationInformation(acceleration: OBUAcceleration) {

    }

    override fun onCardInformation(
        status: OBUCardStatus?,
        paymentMode: OBUPaymentMode?,
        chargingPaymentMode: OBUChargingPaymentMode?,
        balance: Int?
    ) {

    }

    override fun onTravelSummary(travelSummary: OBUTravelSummary) {

    }

    override fun onPaymentHistories(histories: List<OBUPaymentHistory>) {

    }

    override fun onTrafficInformation(trafficInfo: OBUTrafficInfo<out Any>) {

    }

    override fun onError(error: OBUError) {

    }

    override fun onErpChargingAndTrafficInfo(
        trafficInfo: OBUTrafficInfo<out Any>?,
        chargingInfo: List<OBUChargingInformation>?
    ) {

    }
}
```

```

    ) {

    }

    override fun onTotalTripCharged(totalCharged: OBUTotalTripCharged) {

    }

}

OBUSDK.setDataListener(dataListener)

```

1	<b>fun</b> onChargingInformation(chargingInfo: List<OBUCargingInformation>)
	<p>This function provides Electronic Road Pricing information. It returns the list of OBUCargingInformation. The list may contain 2 back-to-back messages. The list may contain the following message combinations:</p> <ul style="list-style-type: none"> <li>i. Charging message + Deduction Successful</li> <li>ii. Charging message + Deduction Failure</li> <li>iii. Entering Priced Road + Pre-Charging</li> <li>iv. Alert Message + Entering pricing road</li> </ul>
2	<b>fun</b> onVelocityInformation(velocity: Double)
	This method provides velocity of the vehicle in every second. The unit is m/s.
3	<b>fun</b> onAccelerationInformation(acceleration: OBUAcceleration)
	This method provides acceleration of the vehicle in every second. The unit is m/s <sup>2</sup> .
4	<b>fun</b> onCardInformation( status: OBUCardStatus?, paymentMode: OBUPaymentMode?, chargingPaymentMode: OBUCargingPaymentMode?, balance: Int? )
	<p>This method provides the Card Status, Payment Mode and Charging Payment Mode, and balance of the CEPAS card.</p> <ul style="list-style-type: none"> <li>i. <b>OBUCardStatus:</b> This represents the status for the CEPAS card inserted in the OBU.</li> <li>ii. <b>OBUPaymentMode:</b> This represents the default payment mode of the OBU.</li> </ul>

	<p>iii. <b>OBUChargingPaymentMode:</b> This represents the instantaneous payment method used for the last deduction. This is only different from the default payment mode if the default payment mode fails, and the OBU falls back to a different payment method, such as the backend payment mode.</p>
5	<p><b>fun</b> onTravelSummary(travelSummary: OBUTravelSummary)</p> <p>This method provides the total travel summary of the last trip. The SDK sends this callback only once after establishing the connection because the travel summary value remains the same as it represents the last trip summary. It contains the following info:</p> <ul style="list-style-type: none"> <li>i. <b>Total Travel Time:</b> The OBU calculates the Total Travel Time from the vehicle engine ignition ON until the ignition OFF. The unit of the value is seconds (sec).</li> <li>ii. <b>Total Travel Distance:</b> The OBU calculates the estimated Total Travel Distance from the vehicle engine ignition ON to ignition OFF. The unit of the value is 10 meters (m).</li> </ul>
6	<p><b>fun</b> onPaymentHistories(histories: List&lt;OBUPaymentHistory&gt;)</p> <p>Payment History is set when charging takes place and the amount is deducted. The OBU always sends five records for the past deductions. If there are no charging all records are set to zero.</p> <p>Note: The payment by car park ticket on EEP without going through OBU is not recorded.</p>
7	<p><b>fun</b> onTrafficInformation(info: OBUTrafficInfo&lt;<b>out</b> Any&gt;)</p> <p>It sends the Traffic Info from the OBU.</p>
8	<p><b>fun</b> onErpChargingAndTrafficInfo(              trafficInfo: OBUTrafficInfo&lt;<b>out</b> Any&gt;?,              chargingInfo: List&lt;OBUChargingInformation&gt;?          )</p> <p>This sends ERP and Traffic info in the same method. It can be used when you want to receive both messages.</p>

9	<b>fun</b> onError(error: OBUError)
	<p>error: the possible errors related to the data</p> <ul style="list-style-type: none"> <li>i. <b>OBUDeviceDataError:</b> Thrown when the raw data is malformed.</li> <li>ii. <b>OBUInternalError:</b> Thrown when an unknown error occurs while processing OBU data.</li> <li>iii. <b>OBUSDKDataAccessPermissionDenied:</b> Thrown when data callback experiences permission not available. This could be the case if the permission was invoked while running the application from the phone's settings while the connection is established.</li> </ul>
10	<b>fun</b> onTotalTripCharged(totalCharged: OBUTotalTripCharged)
	<p>Total charge amount for ERP 2.0 from the ignition ON to OFF. Unit: Singapore cent.</p> <p>Note: The OBU calculates the total charged ERP 2.0 and EEP amounts for the trip from the vehicle engine ignition ON to ignition OFF. ERP1 and EPS charged amounts are not included in the Total Trip Charged amount.</p>

### 3.8.4 SDK Helper Methods for OBU Data

1	<b>fun</b> getCardBalance() : <b>Int?</b>
	This function returns the last card balance if available.
2	<b>fun</b> getPaymentMode() : <b>OBUPaymentMode?</b>
	This function returns the OBU Payment Mode if available.
3	<b>fun</b> getCardStatus() : <b>OBUCardStatus?</b>
	This function returns the card status if available.
4	<b>fun</b> getPaymentHistories() : <b>List&lt;OBUPaymentHistory&gt;?</b>
	This function returns the list of payment histories if available.
5	<b>fun</b> getLastObuData() : <b>OBUData?</b>
	This function returns the last received OBU data.
6	<b>fun</b> getLastObuData() : <b>OBUData?</b>
	This function returns the last received OBU data.
7	<b>fun</b> getPairedDevices(): <b>List&lt;OBU&gt;?</b>
	This function returns a list of paired OBUs.
8	<b>fun</b> getLastTravelSummary() : <b>OBUTravelSummary?</b>
	This function returns the last received Travel Summary data.
9	<b>fun</b> getLastTrafficInfo() : <b>OBUTrafficInfo?</b>
	This function returns the last received Traffic info.
10	<b>fun</b> isConnectionActive() : <b>Boolean</b>
	Returns the connection status.

### 3.8.5 Mock Manager for Android Applications

The Mock Manager is included in the SDK to assist developers in testing their applications without needing an actual OBU device. During development, the Mock Manager emulates a real OBU by providing synthetic OBU functionality (without using Bluetooth to create an actual connection). The Mock Manager can be configured with the following properties:

1. Mock messages sequence: This property sets the mock events for ERP 2.0, ERP1, EEP, EPS, and Traffic messages. The details of Mock events are explained in section 5.1
2. Card balance: This property sets the card balance of the CEPAS card.
3. Message interval: This property sets the time interval for the messages to be sent by the Mock Manager.
4. Cyclic mode: This property displays the messages in an infinite loop.

A mock message simulates a message originating from the mocked OBU. The following is the format for the mock event:

```
MockEvent(List<ERPEvent>,TdcidEvent?)
```

Each mock message can have a list (one or more) electronic events (ERP 2.0, ERP1, EEP, EPS, etc.) and an optional single traffic event. A MockedConnectionHandler object can be created with different messages to mimic real-life situations, and it can be initialised using the code snippet below.

Each ERPEvent relates to OBUChargingInformation and a few fields are configurable from the Mock Manager such as chargeAmount and content. Once these values are configured, you will see them in the OBUChargingInformation object.

The default chargeAmount for MockEvents is 5 cent in Singapore dollars. Please refer to section 5.1 for more details. The charge amount can be set for the ERP events by using the attribute chargeAmount as shown in the sample code below:

```
private val defaultManager =
    MockedConnectionHandler.Builder()
        .setSequence(
            mutableListOf(
                MockEvent(
                    listOf(
                        ERPEvent.PointBasedAlertPointDetected()
                            .also {
                                // $1.00
                                it.chargeAmount = 100
                            }, TdcidEvent.Template110A()
                    ),
                ),
            ),
        )
```

```
MockEvent(
    listOf(
        ERPEvent.PointBasedCharging()
            .also {
                // $1.00
                it.chargeAmount = 100
            }, TdcidEvent.Template110A()
    ),

    MockEvent(listOf(ERPEvent.PointBasedDeductionSuccessful()).also {
        // $1.00
        it.chargeAmount = 100
    }, TdcidEvent.Template110A()
    ))
.setCyclicMode(true)
.setCardBalance(1000) //$10.00
.build()
```

A call to `OBUSDK.disconnect()` exits the cyclic mode.

To enable the Mock Manager, create a `MockedConnectionHandler` object and set in the following method.

```
OBUSDK.enableMockData(mockManager)
```

Once the mock Manager is configured and set to `enableMockSetting()`, calling `start` and `connect` will start mock managers. Implementing the data protocol will send data to all the callbacks. Hard coded values are being used for the following data:

- i. Velocity: The hardcode value is 3 m/s for the mock manager. The actual OBU will send updated velocity of the vehicle after every sec.
- ii. Acceleration: The hardcoded value is (x=0, y=0, z=0) in this `OBUAcceleration`. The actual OBU will send updated acceleration of the vehicle after every sec.
- iii. Trip summary: Total travel time = 30 sec and total travel distance = 500m in this `OBUTripSummary`.
- iv. Trip Charged: The default value is 0 in this `OBUTotalTripCharged`. The actual OBU will send total charge amount for ERP 2.0 from ignition on to off.
- v. Payment histories: The Mock manager sends five payment history objects `OBUPaymentHistory` with empty values. The history will get updated with each charging on the road.



### 3.8.6 Traffic Icons in the OBU SDK

The SDK includes a bundle of icons specifically designed for Traffic messages, which can be utilised by applications. Each TrafficInfo message is associated with an icon that corresponds to a unique icon ID within the SDK. Refer to section 9 for traffic icons list.

The field icon in OBUTrafficInfo (Section 6.7.1) contains the id of the image. The following function in the SDK can be used to retrieve the resource ID:

```
fun getTDCIDImageRes(id: String): Int // return the resource id
```

After getting the resource ID, the getDrawables can be used to obtain the image as shown below:

```
resources.getDrawable(getTDCIDImageRes(icon), null) // get icon from OBUTrafficInfo
```

## 4. iOS

### 4.1 Minimum iOS Version

The SDK supports iOS 13 and higher. Third-party applications are required to set the iOS minimum deployment target to iOS 13.

### 4.2 Programming Language

While the SDK is developed in Swift, the Swift language is recommended for third-party applications. However, it can also be used with Objective-C.

### 4.3 Bluetooth Requirement

The OBU communicates with the SDK via Bluetooth BT 4.1 + LE.

### 4.4 Third-Party Libraries

The SDK is using the following third-party libraries:

1. RealmSwift: it is used to encrypt data using AES-256 encryption.

### 4.5 How to Add the SDK to iOS Applications

The SDK can be installed or updated via CocoaPods. It can be installed from the terminal by running the following command:

```
sudo gem install cocoapods
```

After installing it, run the following:

```
pod init
```

Add the following Pod into the podfile and run the command 'pod install':

```
pod 'extol'
```

## 4.6 SDK Permissions

The SDK communicates with the OBU via Bluetooth and requires Bluetooth access permission. For that, a key [NSBluetoothAlwaysUsageDescription](#) needs to be added to the project's info.plist file. The description tells the user why the application needs access to Bluetooth. For instance:

*“Bluetooth is required to receive ERP and Traffic messages from the OBU”*

Your application will crash if the Info.plist doesn't have this key:  
[NSBluetoothAlwaysUsageDescription](#).

## 4.7 How to Integrate the SDK

The SDK provides `OBUSDKManager`, which is a singleton class, and acts as an entry point of the SDK to call all the methods. It can be accessed as:

`OBUSDKManager.shared`

The SDK for iOS provides a **Handler** that represents either **success** or **failure**, including an associated value in each case.

```
public typealias Handler<T> = (HandlerResult<T, NSError>) -> Void

public enum HandlerResult<Success, Failure> {
    case success(Success)
    case failure(Failure)
}
```

#### 4.7.1 Step 1: Initialise SDK with the SDK Account Key

The first step is to initialise the SDK by entering the key sent to your email after registration at DataMall in the method below. It can be called from the AppDelegate of your application.

```
public func initializeSDK(with sdkKey: String, _ completion: @escaping Handler<Bool>)

// Replace the SDK Account key with your key.
OBUSDKManager.shared.initializeSDK(with: <SDK_ACCOUNT_KEY>) { result in
    switch result {
    case .success(let status):
        print("initialised")
    case .failure(let error):
        print("failed with error code: \(error.code)")
    }
}
```

#### 4.7.2 Step 2: OBU Data Permission

The SDK requires that the developer obtains OBU Data Access permission from the motorist to access OBU data before calling its methods. The developer must ensure that the permission has been granted by the motorist before calling any other methods – as failing to do so will cause the application to crash.

The following method is used to check the status of the permission.

```
// return the status of the permission granted or not
func isDataPermissionGranted() -> Bool
```

This method is used to present the permission alert dialogue to the motorist. If the permission is denied, this method can be called again to request permission. The handler parameter can be used to determine the status of the permission.

```
func requestOBUDataAccessPermission(_ completion: @escaping Handler<Bool>)
```

The sample code is shown below:

```
if ! OBUSDKManager.shared.isDataPermissionGranted() {  
    OBUSDKManager.shared.requestOBUDataAccessPermission { result in  
        switch result {  
            case .success(let status):  
                print("Permission granted")  
            case .failure(let error):  
                print("Permission not granted")  
        }  
    }  
}
```

### 4.7.3 Step 3: Search for OBUs

The SDK provides a delegate to receive search and connection related callbacks. The application must conform to this delegate.

```
OBUSDKManager.shared.connectionStatusDelegate = self
```

This method initiates a search to find nearby OBUs and returns a list of nearby OBUs to the delegate method defined below. The delegate must be implemented before calling the search method.

```
OBUSDKManager.shared.startSearch()
```

This method is triggered when a new device is found during the scanning process, which runs for 60 seconds. The application can listen to the callback and show the list of OBUs to the user for selection.

```
func onOBUFound(_ device: [OBU])
```

#### 4.7.4 Connect to an OBU

The method establishes a connection to the OBU, which can be retrieved from the delegate method mentioned above. Each OBU object contains a name field that can be used to identify it among others.

```
OBUSDKManager.shared.connect(obu: obu)
```

Another way to connect is by using a vehicle number, which can be done by calling the following method. Note that the OBU must be paired with your smartphone for this method to work.

```
func connectWith(vehicleId: String)

OBUSDKManager.shared.connectWith("SME1111Z")
```

#### 4.7.5 Connection Status Delegate

This delegate provides an interface for listening to the search and connection updates. The methods are defined below.

##### 4.7.5.1 OBUConnectionDelegate

1	<b>func</b> onOBUConnected(_device: OBU)
	This method gets called when the connection with the OBU has been established.
2	<b>func</b> onOBUDisconnected(_device: OBU, error: NSError?)  This method gets called when the connection with the OBU has been disconnected. The disconnection could occur due to internal Bluetooth disruptions or the OBU going out of Bluetooth range. The error object contains the error from the core Bluetooth framework.
3	<b>func</b> onOBUFound(_ device: [OBU])  The startSearch() method returns a list of nearby OBUs, and the device continues scanning until the search is stopped after 60 seconds.
4	<b>func</b> onOBUConnectionFailure(_ error: NSError)

	<p>This function is called when the connection fails. It could fail due to the following reasons:</p> <ul style="list-style-type: none"> <li>i. <b>INTERNET_NOT_AVAILABLE:</b> An Internet connection is required by the SDK to fetch OBU related data from the server. In case if there is not internet connectivity, this error is returned.</li> <li>ii. <b>CONNECTION_FAILED:</b> when the connection between the application and OBU is not successful. It could be due to Bluetooth disruption or Bluetooth out of range.</li> <li>iii. <b>MAC_ADDRESS_AUTHENTICATION_FAILED:</b> OBU authenticate your application against the MAC Address of your iPhone. Please ensure your iPhone is registered at 'e-Services' portal. The same Mac Address registered with the OBU must be keyed in the application. The Mac Address authentication screen is presented from the SDK during the connection process. Once added, the SDK will not show it again.</li> <li>iv. <b>BLUETOOTH_DISABLED:</b> when the Bluetooth is disabled.</li> <li>v. <b>NO_PUBLIC_KEY_AVAILABLE:</b> This is an internal error. If this occurs, make sure that your phone is registered with e-Services portal for your OBU,</li> <li>vi. <b>CODE_SIGNATURE_FAIL:</b> This is an internal SDK error. If this error persists, please contact the customer support.</li> <li>vii. <b>OBU_NOT_PAIRIED:</b> When connecting using vehicle number with <b>func connectWith(vehicleId: String)</b>, this error is thrown if the OBU registered with the Vehicle number has not been paired with the phone.</li> <li>viii. <b>OBU_DATA_ACCESS_PERMISSION_DENIED:</b> This is returned if the OBU data permission has not been taken from the motorist. Upon Receiving this, the developer needs to call the function <code>requestOBUDataAccessPermission()</code>.</li> </ul>
5	<p><b>func onBluetoothStateUpdated(_ state: BluetoothState)</b></p> <p>This function is called when the CBCentralManager changes its state. The states are as follows:</p> <ul style="list-style-type: none"> <li>i. Unknown</li> <li>ii. Resetting</li> <li>iii. Unsupported</li> <li>iv. Unauthorized</li> <li>v. Powered Off</li> <li>vi. Powered On</li> </ul>
6	<p><b>func onSearchFinished()</b></p> <p>The search runs for 60 seconds. This is a helper function to get the update that the search has stopped. The application can use this info to show a message to the user.</p>

Once the connection has been established, the following data delegates are used to receive data from the OBU.

#### 4.7.5.2 OBUDataDelegate

The Data delegate provides an interface for receiving data items from the OBU after the connection. The application needs to implement the delegate below.

OBUSDKManager.shared.dataDelegate = self

1	<b>func</b> onVelocityInformation(_ velocity: Double)
	This function provides velocity info of the vehicle in meter/second.
2	<b>func</b> onAccelerationInformation(_ acceleration: OBUAcceleration)
	This function provides acceleration info of the vehicle.
3	<b>func</b> onChargingInformation(_ chargingInfo: [OBUChargingInformation])
	This function provides charging info from the OBU.
4	<b>func</b> onTrafficInformation(_ trafficInfo: OBUTrafficData)
	This function provides traffic info from the OBU.
5	<b>func</b> onErpChargingAndTrafficInfo(trafficInfo: OBUTrafficData, chargingInfo: [OBUChargingInformation]?)
	This function provides charging & traffic message from the OBU. This method is invoked on certain points on the road.
6	<b>func</b> onPaymentHistories(_ histories: [OBUPaymentHistory])
	This function always returns the last five charging objects from the OBU for ERP. In case of less than five charging, the remaining objects will be empty but there will be always five objects.
7	<b>func</b> onTravelSummary(_ travelSummary: OBUTravelSummary)
	<p>This method provides the total travel summary of the trip. The SDK send this callback only once after establishing the connection because the summary/value will not change, as it represents the last trip summary. It contains the following:</p> <ul style="list-style-type: none"> <li>i. <b>Total Travel Time:</b> The OBU calculates the Total Travel Time from the vehicle engine ignition ON until the ignition OFF. The unit of the value is seconds (sec).</li> </ul>



	<p>ii. <b>Total Travel Distance:</b> The OBU calculates the estimated Total Travel Distance from the vehicle engine ignition ON to ignition OFF. The unit of the value is 10 meters (m).</p>
8	<p><b>func onTotalTripCharged(_ totalCharged: OBUTotalTripCharged)</b></p> <p>Total charge amount for ERP from the ignition ON to OFF.</p> <p>Unit: Singapore cent.</p> <p>Note: The OBU calculates the total charged ERP 2.0 and EEP amounts for the trip from the vehicle engine ignition ON to ignition OFF. ERP1 and EPS charged amounts are not included in the Total Charged amount.</p>
9	<p><b>func onError(_ errorCode: NSError)</b></p> <p>This function provides the following errors:</p> <p>i. <b>DEVICE_DATA_ACCESS_ERROR:</b> This is returned when the data is malformed. Please contact customer support if it persists.</p>
10	<p><b>func onCardInformation(_ status: OBUCardStatus?, paymentMode: OBUPaymentMode?, chargingPaymentMode: OBUCargingPaymentMode?, balance: Int)</b></p> <p>This function provides the following info of the inserted card:</p> <p>i. Card status.</p> <p>ii. Payment mode: if CEPAS card is inserted, it returns frontend else backend.</p> <p>iii. Charging payment mode: this reflects the mode the OBU used to deduct the payment. For instance, if a CEPAS card is inserted but balance is not sufficient, the deduction will be made on the backend and this will return backend payment mode in the method. This info can be reflected to the motorist by using this method.</p> <p>iv. Current balance of the card: Balance of the inserted CEPAS card.</p>

## 4.7.6 SDK Helper Methods for OBU Data

1	<b>func</b> <code>getCardBalance()</code> -> <code>Int?</code>
	This function returns the last card balance if available (S\$ cents).
2	<b>func</b> <code>getPaymentMode()</code> -> <code>OBUPaymentMode?</code>
	This function returns the payment mode if available.
3	<b>func</b> <code>getCardStatus()</code> -> <code>OBUCardStatus?</code>
	This function returns the card status if available.
4	<b>func</b> <code>getPaymentHistories()</code> -> <code>[OBUPaymentHistory]?</code>
	This function returns the payment histories.
5	<b>func</b> <code>getLastObuData()</code> -> <code>OBUData?</code>
	This function returns the last received OBU Data. This class contains all the OBU data. It can be used to get complete data in one method.
6	<b>func</b> <code>getPairedDevices(_ completion: @escaping Handler&lt;[OBU]?&gt;)</code>
	This function returns a list of paired OBUs.
7	<b>func</b> <code>isSDKInitialised()</code> -> <code>Bool</code>
	Returns the SDK initialised status.
8	<b>func</b> <code>isDataPermissionGranted()</code> -> <code>Bool</code>
	Returns the OBU Data permission status.
9	<b>func</b> <code>getLastTrafficData()</code> -> <code>OBUTrafficInfo?</code>
	Returns the last received traffic info data.
10	<b>func</b> <code>getLastTravelSummary()</code> -> <code>OBUTravelSummary?</code>
	Returns the last received Travel Summary data.

## 4.7.7 Mock Manager for iOS Applications

The Mock Manager is included in the SDK to assist developers in testing their applications without needing an actual OBU device. During development, the Mock Manager emulates a real OBU by providing synthetic OBU functionality (without using Bluetooth to create an actual connection). The Mock Manager can be configured with the following properties:

1. Mock messages sequence: This property sets the mock events for ERP 2.0, ERP1, EEP, EPS, and Traffic messages. The details of Mock events are explained in section 5.1
2. Card balance: This property sets the card balance of the CEPAS card.
3. Message interval: This property sets the time interval for the messages to be sent by the Mock Manager.
4. Cyclic mode: This property displays the messages in an infinite loop.

The iOS OBU SDK provides a class named **MockSetting**, which can be used to configure the mock messages through the builder pattern.

1	<code>var sequence: [MockEvent]</code>
	<p>The OBU can send two types of messages: ERP (Electronic Road Pricing) and Traffic messages. ERP messages will be sent at specific points based on the type (Point based) mentioned in section. OBU can send either ERP/TDCID or both ERP and TDCID. These scenarios can be tested by choosing the mock events from section 5.1 .</p> <p>electronicEventList: [ERPEvent] tdcidEvent: TdcidEvent?</p> <p>The ERP messages is of type list meaning it can have multiple messages at the same time. It is recommended to take the last item if multiple messages are received i.e. count &gt; 1.</p> <p>The ERP 2.0/ERP1/EEP/EPS messages can be selected from the list mentioned above. The TDCID messages can be selected from the list mentioned above.</p>
2	<code>var cyclicMode: Bool = false</code>
	Boolean value for receiving infinite messages in the loop. It will keep repeating the sequence. The default value is false.
3	<code>var timeInterval: Int = 5</code>
	Time interval of the messages. The default value is 5 seconds.
4	<code>var cardBalance: Int</code>
	This value is in S\$ cents.

Each ERPEvent relates to OBUChargingInformation and a few fields are configurable from the Mock Manager such as chargeAmount and content. Once these values are configured, you will see them in the OBUChargingInformation object.

The default chargeAmount for ERPEvents is 5 cent in Singapore dollars. Please refer to section 5.1 for more details

Code Snippet:

```
// Step 1:
let events = ([MockEvent(electronicEventList: [PointBasedAlertPointDetected(chargeAmount:
100)], tdcidEvent: nil), MockEvent(electronicEventList: [PointBasedCharging(chargeAmount:
100)], tdcidEvent: nil), MockEvent(electronicEventList:
[PointBasedDeductionSuccessful(chargeAmount:100)], tdcidEvent: nil)]

// Step 2:
let settings = MockBuilder()
    .setCardBalance(Int(1000)) // $10.00
    .setTimeInterval(2) // 2 sec
    .setCyclicMode(true)
    .setSequence(events)
    .build()

// Step 3:
OBUSDKManager.shared.enableMockSetting(setting)
```

Once the mock Manager is configured and set to enableMockSetting(), calling start and connect will start mock managers. Implementing the data protocol will enable sending data to the callbacks. The following hard-coded values are being used for the data:

- i. Velocity: The hardcode value is 3 for the mock manager. The actual OBU will send updated velocity of the vehicle after every sec.
- ii. Acceleration: The hardcoded value is (x=0, y=0, z=0) in this OBUAcceleration. The actual OBU will send updated acceleration of the vehicle after every sec.
- iii. Trip summary: Total travel time = 30sec and total travel distance = 500m in this OBUTripSummary.
- iv. Trip Charged: The default value is 0 in this OBUTotalCharged. The actual OBU will send total charge amount for ERP 2.0 from ignition on to off.
- v. Payment histories: The Mock manager sends five payment history objects OBUPaymentHistory with empty values. The history will get updated with each charging on the road.

#### 4.7.8 Traffic Icons in SDK for iOS

The SDK includes a bundle of icons specifically designed for Traffic messages, which can be utilised by applications. Each TrafficInfo message is associated with an icon that corresponds to a unique icon ID within the SDK. Refer to Appendix A for traffic icon list.

The SDK has the bundle ID 'sg.gov.lta.extol' and the bundle name 'TDCID'. The code is shown below:

```
if let bundle = Bundle(identifier: "sg.gov.lta.extol")?.path(forResource: "TDCID", ofType:
"bundle") {
    if let imagePath = Bundle(path: bundle)?.path(forResource: "1402", ofType:
"png") {
        print("image:\(imagePath)")
        iconView.image = UIImage.init(contentsOfFile: imagePath)
    }
}
```

## **5. MOCK MANAGER FOR TESTING APPLICATIONS**

The SDK is shipped with the Mock Manager as a way to test the applications by providing a set of APIs to generate mock data which is an exact replica of the messages sent from the OBU. The Mock Manager is used to test the following messages:

1. Electronic Road Pricing Messages (ERP)
2. Electronic Enhanced Parking (EEP)
3. Electronic Parking System (EPS)
4. Traffic Messages

The SDK provides the following mock events for testing the applications:

### **5.1 Electronic Road Pricing (ERP)**

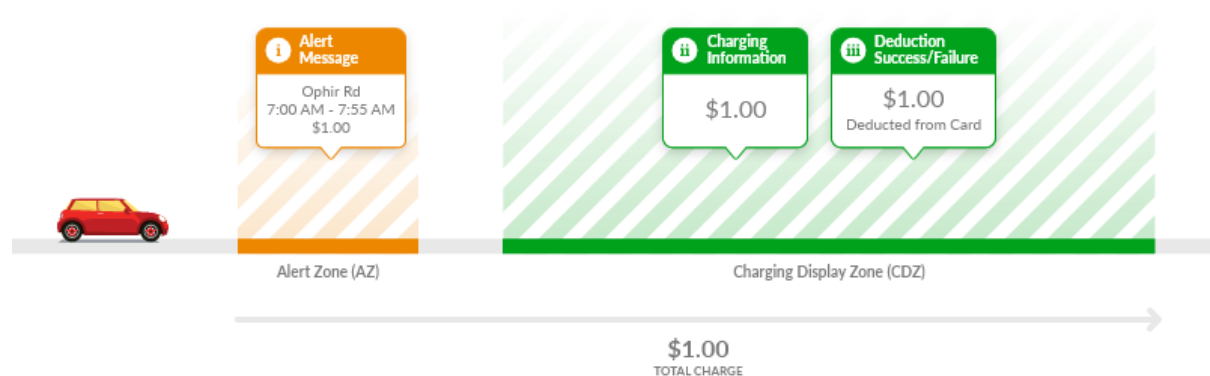
The following section explains a number of events for each charging type. The event names are actual classes/Enums that can be set in the mock manager explained in this guide. Setting up the mock manager will simulate the OBU behaviour and generate the messages that the OBU will generate while driving on the roads.

All ERP 2.0/ERP1/EEP/EPS messages data is sent in `OBUChargingInformation` object. There is a default `chargeAmount` 5 cent in Singapore dollars for all the ERP related events. The developer can also configure this for each `ERPEvent`.

### 5.1.1 Point-Based Charging

The OBU sends these events for Point-Based Charging.

- i. PointBasedAlertPointDetected: when the vehicle enters alert zone. The content1 is configurable for this message – which is used to show the road name as shown in the figure below for this message.
- ii. PointBasedCharging: when the vehicles go near to the charging zone. The charge Amount is configurable.
- iii. PointBasedDeductionSuccessful: upon Successful charging. The charge Amount is configurable.
- iv. PointBasedDeductionFailure: when the charging deduction fails.



### 5.1.2 ERP1-Based Charging

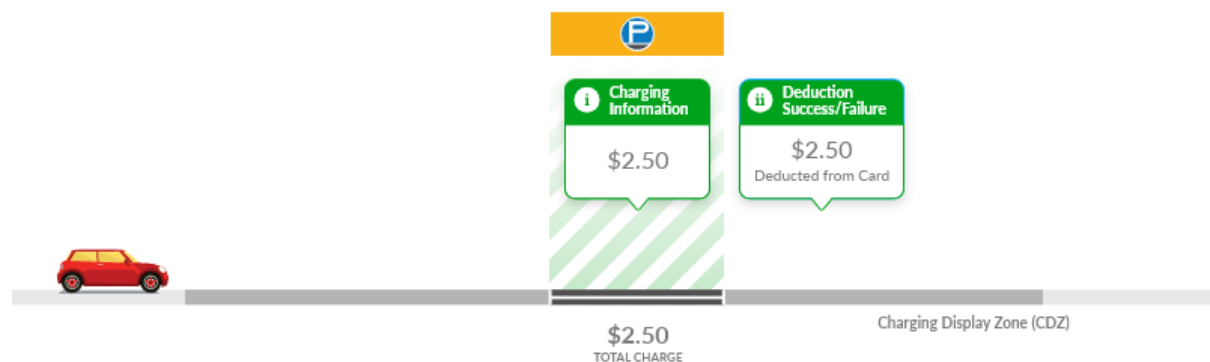
- i. ERP1BasedCharging: when the vehicle approaches the Charging Display Zone.
- ii. ERP1BasedDeductionSuccessful: upon Successful charging.
- iii. ERP1BasedDeductionFailure: when the charging deduction fails.

### 5.1.3 Common Alert

- i. CommonAlertPointDetected: This contains a road message. The sample shown in section 1.5. The content1 and content2 are configurable for this message. The default values are 'After Exit 11' and '(Paya Lebar Rd)'.

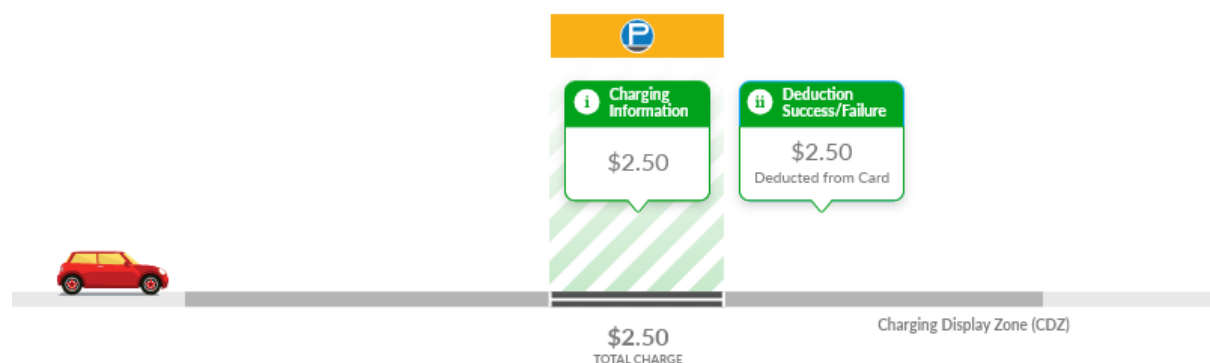
### 5.1.4 Electronic Enhanced Parking (EEP)

- ii. EEPInfo: This information is available in content1,2,3,4 fields of OBUCoinInformation explained in Section 6.3.
- iii. EEPBasedCharging: when the vehicle approaches the Charging Display Zone.
- iv. EEPBasedDeductionSuccessful: upon Successful charging.
- v. EEPBasedDeductionFailure: when the charging deduction fails.



### 5.1.5 Electronic Parking System (EPS)

- i. EPSBasedCharging: when the vehicle approaches the Charging Display Zone.
- ii. EPSBasedDeductionSuccessful: upon Successful charging.
- iii. EPSBasedDeductionFailure: when the charging deduction fails.





## 5.1.6 Traffic Messages

The Traffic messages/events are divided into three types:

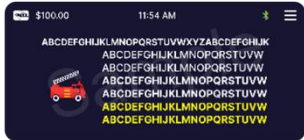
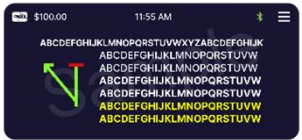
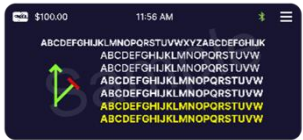
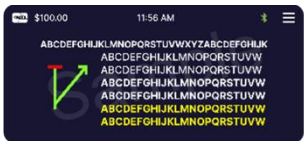
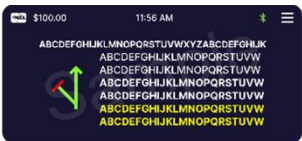
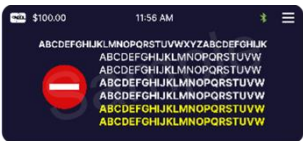
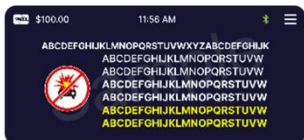
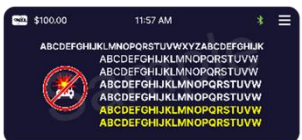
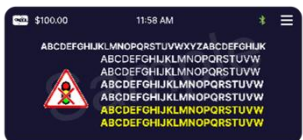
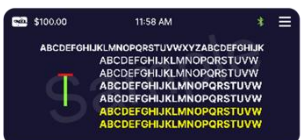
- Traffic Text
- Travel Time
- Parking lots information.

The screenshots of these events are shown below to provide an example of how the information for these messages could be used in your application. The messages are represented in the class `OBUTrafficInfo`. It contains an attribute 'templateId' which represents the type of messages. The list below shows a type of messages that are used with the Mock Manager.

### 5.1.6.1 Template1

This template is defined as **unplannedIncidentsType1** and used to represent general traffic information. This belongs to type **Traffic Text**. The following are the list of templates and their corresponding messages and display layout.

<p>Template1A</p> 	<p>Template1B</p> 	<p>Template1C</p> 
<p>Template1D</p> 	<p>Template1E</p> 	<p>Template1F</p> 
<p>Template1G</p> 	<p>Template1H</p> 	<p>Template1I</p> 
<p>Template1J</p> 	<p>Template1K</p> 	<p>Template1L</p> 

<p>Template1M</p> 	<p>Template1N</p> 	<p>Template1O</p> 
<p>Template1P</p> 	<p>Template1Q</p> 	<p>Template1R</p> 
<p>Template1S</p> 	<p>Template1T</p> 	<p>Template1U</p> 
<p>Template1V</p> 		

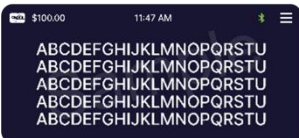

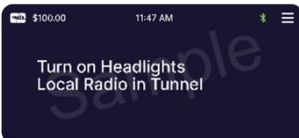
### 5.1.6.2 Template2

This template is defined as **unplannedIncidentsTypell** and used to represent general traffic information. This belongs to type **Traffic Text**. The following are the list of templates and their corresponding messages and display layout.

<p>Template2A</p> 	<p>Template2B</p> 	<p>Template2C</p> 
<p>Template2D</p> 	<p>Template2E</p> 	

### 5.1.6.3 Template3

This template is defined as **freeText5Line** and used to represent general traffic information. This belongs to type **Traffic Text**. There can be maximum five lines. The following are the list of templates and their corresponding messages and display layout.:

<p>Template3A</p> 	<p>Template3B</p> 	<p>Template3C</p> 
---	---	---

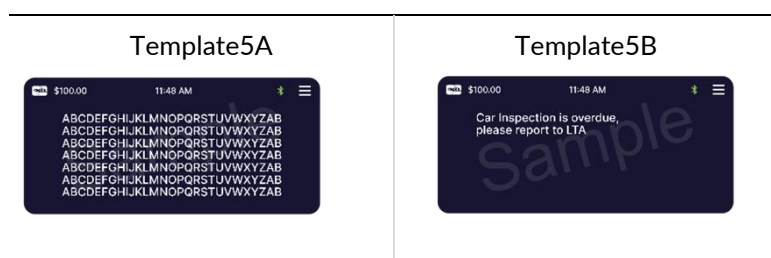
### 5.1.6.4 Template4

This template is defined as **freeText6Line** and used to represent general traffic information. This belongs to type **Traffic Text**. There can be maximum six lines. The following are the list of templates and their corresponding messages and display layout.



### 5.1.6.5 Template5

This template is defined as **freeText7Line** and used to represent general traffic information. This belongs to type **Traffic Text**. There can be maximum seven lines. The following are the list of templates and their corresponding messages and display layout.



### 5.1.6.6 Template6

This template is defined as **freeText8Line** and used to represent general traffic information. This belongs to type **Traffic Text**. This template can display a maximum eight lines. The following are the list of templates and their corresponding messages and display layout.




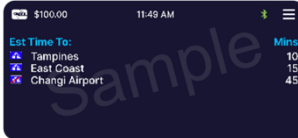
### 5.1.6.7 Template8

This template is defined as **plannedIncidents** and used to represent traffic information. This belongs to type **Traffic Text**. The following are the list of templates and their corresponding messages and display layout.

<p>Template8A</p>	<p>Template8B</p>	<p>Template8C</p>
<p>Template8D</p>	<p>Template8E</p>	<p>Template8F</p>
<p>Template8G</p>	<p>Template8H</p>	<p>Template8I</p>
<p>Template8J</p>	<p>Template8K</p>	<p>Template8L</p>
<p>Template8M</p>	<p>Template8N</p>	<p>Template8O</p>
<p>Template8P</p>	<p>Template8Q</p>	

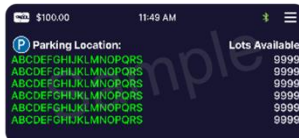
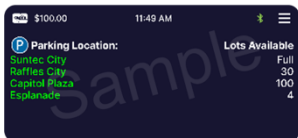
### 5.1.6.8 Template110

This **traveltime** template provides the information of the estimated time required to reach the destinations. The following are the list of templates and their corresponding messages and display layout.

Template110A	Template110B																				
 <p>Estimated travel times for various destinations:</p> <table border="1"> <thead> <tr> <th>Est Time To:</th> <th>Mins</th> </tr> </thead> <tbody> <tr><td>ABCDEFGHIJKLMNQRST</td><td>9999</td></tr> <tr><td>ABCDEFGHIJKLMNQRST</td><td>9999</td></tr> <tr><td>ABCDEFGHIJKLMNQRST</td><td>9999</td></tr> <tr><td>ABCDEFGHIJKLMNQRST</td><td>9999</td></tr> <tr><td>ABCDEFGHIJKLMNQRST</td><td>9999</td></tr> </tbody> </table>	Est Time To:	Mins	ABCDEFGHIJKLMNQRST	9999	ABCDEFGHIJKLMNQRST	9999	ABCDEFGHIJKLMNQRST	9999	ABCDEFGHIJKLMNQRST	9999	ABCDEFGHIJKLMNQRST	9999	 <p>Estimated travel times for specific locations:</p> <table border="1"> <thead> <tr> <th>Est Time To:</th> <th>Mins</th> </tr> </thead> <tbody> <tr><td>Tampines</td><td>10</td></tr> <tr><td>East Coast</td><td>15</td></tr> <tr><td>Changi Airport</td><td>45</td></tr> </tbody> </table>	Est Time To:	Mins	Tampines	10	East Coast	15	Changi Airport	45
Est Time To:	Mins																				
ABCDEFGHIJKLMNQRST	9999																				
ABCDEFGHIJKLMNQRST	9999																				
ABCDEFGHIJKLMNQRST	9999																				
ABCDEFGHIJKLMNQRST	9999																				
ABCDEFGHIJKLMNQRST	9999																				
Est Time To:	Mins																				
Tampines	10																				
East Coast	15																				
Changi Airport	45																				

### 5.1.6.9 Template111

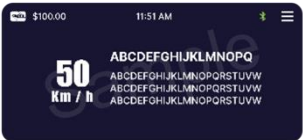
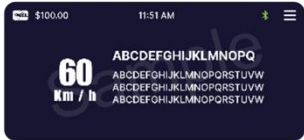
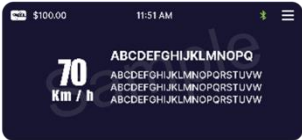
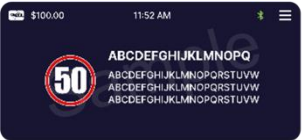
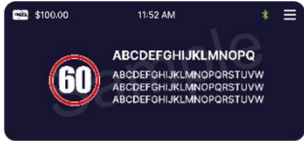
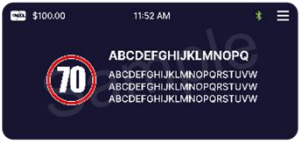
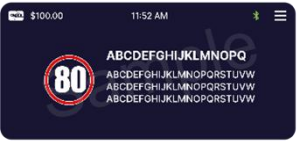
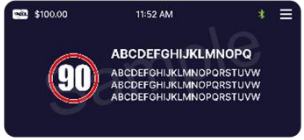
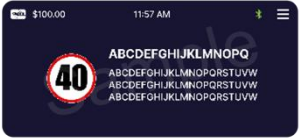
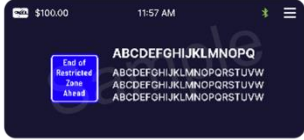
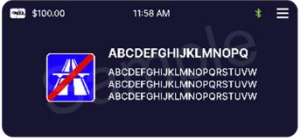
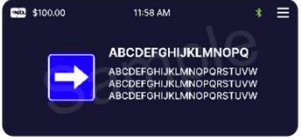
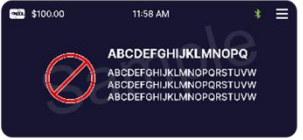
This parking template defines and provides the information for the available parking lots. The following are the list of templates and their corresponding messages and display layout.

Template111A	Template111B																						
 <p>Available parking lots:</p> <table border="1"> <thead> <tr> <th>Parking Location:</th> <th>Lots Available</th> </tr> </thead> <tbody> <tr><td>ABCDEFGHIJKLMNQRST</td><td>9999</td></tr> <tr><td>ABCDEFGHIJKLMNQRST</td><td>9999</td></tr> <tr><td>ABCDEFGHIJKLMNQRST</td><td>9999</td></tr> <tr><td>ABCDEFGHIJKLMNQRST</td><td>9999</td></tr> <tr><td>ABCDEFGHIJKLMNQRST</td><td>9999</td></tr> </tbody> </table>	Parking Location:	Lots Available	ABCDEFGHIJKLMNQRST	9999	ABCDEFGHIJKLMNQRST	9999	ABCDEFGHIJKLMNQRST	9999	ABCDEFGHIJKLMNQRST	9999	ABCDEFGHIJKLMNQRST	9999	 <p>Available parking lots with specific locations:</p> <table border="1"> <thead> <tr> <th>Parking Location:</th> <th>Lots Available</th> </tr> </thead> <tbody> <tr><td>Series City</td><td>Full</td></tr> <tr><td>Raffles City</td><td>30</td></tr> <tr><td>Capitol Plaza</td><td>100</td></tr> <tr><td>Esplanade</td><td>4</td></tr> </tbody> </table>	Parking Location:	Lots Available	Series City	Full	Raffles City	30	Capitol Plaza	100	Esplanade	4
Parking Location:	Lots Available																						
ABCDEFGHIJKLMNQRST	9999																						
ABCDEFGHIJKLMNQRST	9999																						
ABCDEFGHIJKLMNQRST	9999																						
ABCDEFGHIJKLMNQRST	9999																						
ABCDEFGHIJKLMNQRST	9999																						
Parking Location:	Lots Available																						
Series City	Full																						
Raffles City	30																						
Capitol Plaza	100																						
Esplanade	4																						



### 5.1.6.10 Template12

This template is defined as **preloadedData**. This belongs to type **Traffic Text**. The following are the list of templates and their corresponding messages and display layout.

<p>Template12A</p> 	<p>Template12B</p> 	<p>Template12C</p> 
<p>Template12D</p> 	<p>Template12E</p> 	<p>Template12F</p> 
<p>Template12G</p> 	<p>Template12H</p> 	<p>Template12I</p> 
<p>Template12J</p> 	<p>Template12K</p> 	<p>Template12L</p> 
<p>Template12M</p> 	<p>Template12N</p> 	<p>Template12O</p> 
<p>Template12P</p> 	<p>Template12Q</p> 	<p>Template12R</p> 

### 5.1.6.11 Template13

This template is defined as **imageDisplay**. This belongs to type **Traffic Text**. The following are the list of templates and their corresponding messages and display layout.





## 6. API CLASSES

These are the classes used in the APIs for iOS and Android.

### 6.1 OBU

This represents the OBU object containing the name of an OBU.

PROPERTIES	DESCRIPTION
<b>name</b>	<b>String</b> Name of the OBU.

### 6.2 OBUAcceleration

This set of parameters defines the acceleration data. The unit is m<sup>2</sup>/s.

PROPERTIES	DESCRIPTION
<b>x</b>	<b>Float</b> Acceleration of a vehicle along X axis
<b>y</b>	<b>Float</b> Acceleration of a vehicle along Y axis
<b>z</b>	<b>Float</b> Acceleration of a vehicle along Z axis

## 6.3 OBUChargingInformation

This set of parameters defines the OBU's charging information types:

PROPERTIES	DESCRIPTION
<b>businessFunction</b>	<b>OBUBusinessFunction</b> Business function of the charging type. Currently ERP, EEP and EPS are available in the API
<b>chargingMessageType</b>	<b>OBUChargingMessageType</b> Type of the message sent from the OBU
<b>chargingType</b>	<b>OBUChargingType</b> Charging type of the OBU (Point based)
<b>paymentMode</b>	<b>OBUChargingPaymentMode</b> Payment mode of the OBU. It could be frontend (CEPAS) or backend (card enabled at e-Services portal).
<b>cardStatus</b>	<b>OBUCardStatus</b> Card Status
<b>content1</b>	<b>String?</b> Represent additional info to be used for Common alert, CPO and EEP info
<b>content2</b>	<b>String?</b> Represent additional info to be used for Common alert, CPO and EEP info
<b>content3</b>	<b>String?</b> Represent additional info to be used for CPO and EEP info
<b>content4</b>	<b>String?</b> Represent additional info to be displayed for CPO and EEP Info
<b>roadName</b>	<b>String?</b> This value is used for Common alert message
<b>chargingAmount</b>	<b>Int</b> Charging rate to be deducted for charging messages. The amount is in Singapore dollars in cents.
<b>cardBalance</b>	<b>Int</b> Balance of inserted CEPAS card in Singapore dollar cents

## 6.4 OBUPaymentHistory

This class represents payment history for successful charging.

PROPERTIES	DESCRIPTION
<b>sequentialNumber</b>	<b>Int</b> Id of the payment object
<b>paymentDate</b>	<b>String?</b> Represents date and time when the charging occurred. (format: YYYYMMDDHHMMSS)
<b>businessFunction</b>	<b>OBUBusinessFunction</b> Business function of the OBU (ERP/EPS/EEP/OPC/CPT/REP)  *OPC/CPT/REP are reserved for future
<b>paymentMode</b>	<b>OBUChargingPaymentMode</b> Charging Payment mode at the time of charging (Backend or Frontend)
<b>chargeAmount</b>	<b>Int</b> The charged amount after successful deduction in cents

## 6.5 OBUTravelSummary

The OBU creates a Travel Summary when the vehicle's ignition is turned off. When the ignition turns on again, it sets the total travel distance of the previous trip and sent data to this class.

PROPERTIES	DESCRIPTION
<b>totalTravelTime</b>	<b>Short</b> Total Travel time covered in the last journey (Unit: Sec)
<b>totalTravelDistance</b>	<b>Short</b> Total travel distance covered in the last journey (Unit: 10m)

## 6.6 OBUTotalTripCharged

This class contains total charging info for all the business functions. The unit is cent.

PROPERTIES	DESCRIPTION
<b>totalERP2charged</b>	<b>Int</b> Total charge amount for ERP 2.0 from ignition ON to OFF.
<b>totalEEPCharged</b>	<b>Int</b> Total charge amount for EEP from ignition ON to OFF
<b>totalREPCharged</b>	<b>Int</b> Total charge amount for REP from ignition ON to OFF
<b>totalOPCharged</b>	<b>Int</b> Total charge amount for OPC from ignition ON to OFF
<b>totalCPTCharged</b>	<b>Int</b> Total charge amount for CPT from ignition ON to OFF

## 6.7 Traffic Information

Traffic messages are represented by the `OBUTrafficInfo` model. There are 3 types of traffic messages. The following are the variants:

1. `TrafficParking`: This represents parking information, i.e. number of parking spots available at different places.
2. `TravelTime`: It provides information about travel time to certain places.
3. `TrafficText`: It provides information about traffic related messages.

### 6.7.1 Usage in Android

In Android a sealed class is being used to provides the three types of Traffic messages.

```
sealed class OBUTrafficInfo<T>(  
    val priority: OBUTrafficDataPriority, val templateld: OBUTrafficMessageType,  
    val dataList: List<T>, val icon: OBUIImageRef?  
) {  
  
    class OBUTrafficParking(  
        priority: OBUTrafficDataPriority,  
        templateld: OBUTrafficMessageType,  
        dataList: List<Parking>,  
        icon: OBUIImageRef?  
    ) : OBUTrafficInfo<Parking>(priority, templateld, dataList, icon)  
  
    class OBUTrafficTravelTime(  
        priority: OBUTrafficDataPriority,  
        templateld: OBUTrafficMessageType,  
        dataList: List<TravelTime>,  
        icon: OBUIImageRef?  
    ) : OBUTrafficInfo<TravelTime>(priority, templateld, dataList, icon)  
  
    class OBUTrafficText(  
        priority: OBUTrafficDataPriority,  
        templateld: OBUTrafficMessageType,  
        dataList: List<OBUTextWithStyle>,  
        icon: OBUIImageRef?  
    ) : OBUTrafficInfo<OBUTextWithStyle>(priority, templateld, dataList, icon)
```

## 6.7.2 Usage in iOS

In iOS, an enum `OBUTrafficInfo` is used to send the three types of Traffic messages:

```
public protocol OBUBaseTraffic {
    associatedtype T
    var priority: OBUTrafficDataPriority { get set }
    var templateId: OBUTrafficMessageType { get set }
    var dataList: [T]? { get set }
}

public struct OBUTrafficParking: OBUBaseTraffic {
    public var priority: OBUTrafficDataPriority
    public var templateId: OBUTrafficMessageType
    public var dataList: [Parking]?
    public typealias T = Parking
}

public struct OBUTrafficTime: OBUBaseTraffic {
    public var priority: OBUTrafficDataPriority
    public var templateId: OBUTrafficMessageType
    public var dataList: [TravelTime]?
    public typealias T = TravelTime
}

public struct OBUTrafficText: OBUBaseTraffic {
    public var priority: OBUTrafficDataPriority
    public var templateId: OBUTrafficMessageType
    public var dataList: [OBUTextWithStyle]?
    public typealias T = OBUTextWithStyle
    public let icon: OBUIImageRef?
}

@frozen public enum OBUTrafficInfo {
    case trafficText(OBUTrafficText)
    case trafficParking(OBUTrafficParking)
    case travelTime(OBUTrafficTime)
}
```

The classes used in the `OBUTrafficInfo` are shown in tables below:

### 6.7.2.1 Parking

PROPERTIES	DESCRIPTION
<b>location</b>	<b>OBUTextWithStyle</b> Represents the parking lots name.
<b>lots</b>	<b>OBUTextWithStyle</b> Represent the number of lots available.

### 6.7.2.2 TravelTime

PROPERTIES	DESCRIPTION
<b>location</b>	<b>OBUTextWithStyle</b> Represents the destination name.
<b>min</b>	<b>OBUTextWithStyle</b> Represents the minimum time to reach the location.
<b>icon</b>	<b>OBUIImageRef</b> It is a typealias of String and contains the icon name to be shown for the TDCID messages.

### 6.7.2.3 OBUTextWithStyle

The class provides the text information for each line in the Traffic info data.

PROPERTIES	DESCRIPTION
<b>text</b>	<b>String</b> Represents the data shown for TrafficInfo messages.
<b>color</b>	<b>OBUTextColor</b> Represent the color of the text OBU display shows for each text for traffic messages
<b>style</b>	<b>OBUTextStyle</b> Represents the font weight of each text whether its normal or Bold shown by the OBU Display

## 6.8 OBUData

This class contains all data points in one place.

PROPERTIES	DESCRIPTION
<b>velocity</b>	<b>Double</b> Velocity of the vehicle
<b>acceleration</b>	<b>OBUAcceleration?</b> Acceleration of the vehicle
<b>chargingInformations</b>	<b>OBUChargingInformation?</b> A list of Charging Information
<b>paymentHistories</b>	<b>OBUPaymentHistory?</b> Last five objects of Charging Payment History
<b>cardBalance</b>	<b>Int?</b> Balance of inserted CEPAS Card
<b>cardStatus</b>	<b>OBUCardStatus?</b> Status of inserted CEPAS Card
<b>paymentMode</b>	<b>OBUPaymentMode?</b> Payment mode of the OBU
<b>trafficInformation</b>	<b>OBUTrafficInfo?</b> Traffic Data
<b>travelSummary</b>	<b>OBUTravelSummary?</b> Travel Summary
<b>totalCharged</b>	<b>OBUTotalTripCharged?</b> Total Trip Charged



## 6.9 Enums

### 6.9.1 OBUCardStatus

VALUES	DESCRIPTION
<b>Detected</b>	It represents initial value of the inserted CEPAS card
<b>Blacklisted</b>	It represents Blacklisted card
<b>ExpiredStored</b>	It represents Expired stored value card
<b>Invalid</b>	Invalid card
<b>IssuerIDError</b>	Card Issuer ID Error
<b>NoCardForFrontendPayment</b>	No card or front-end payment mode
<b>Blocked</b>	It represents blocked card
<b>FaultyStoreValue</b>	Faulty stored value card
<b>InsufficientStoredValue</b>	Insufficient Stored Value
<b>WrongStoredValueDebitCertificate</b>	Wrong Stored Value Debit Certificate
<b>NFCAccessError</b>	NFCAccessError

### 6.9.2 OBUBusinessFunction

VALUES	DESCRIPTION
<b>None</b>	None
<b>ERP</b>	Electronic Road Pricing
<b>EEP</b>	Electronic Enhanced Paring
<b>EPS</b>	Electronic Parking System
<b>REP</b>	Reserved for future
<b>OPC</b>	Reserved for future
<b>CPT</b>	Reserved for future

### 6.9.3 OBUCargingMessageType

VALUES	DESCRIPTION
<b>None</b>	None
<b>AlertPoint</b>	Alert point for ERP (Point-based charging)
<b>Charging</b>	Pre-Charging message for all business functions
<b>DeductionSuccessful</b>	Deduction Successful message for all business function
<b>DeductionFailure</b>	Deduction Failure message for all business function
<b>CPOInformation</b>	CPO Information for EEP
<b>Common</b>	Common Alert

### 6.9.4 OBUCargingPaymentMode

VALUES	DESCRIPTION
<b>None</b>	None
<b>Frontend</b>	OBU using CEPAS card for ERP
<b>Backend</b>	OBU using backend payment method enabled at e-Services portal.

### 6.9.5 OBUCargingType

VALUES	DESCRIPTION
<b>None</b>	None
<b>PointBased</b>	Point-based charging
<b>Common</b>	Common Alert message type
<b>Erp1</b>	ERP1 message

## 6.9.6 OBUPaymentMode

VALUES	DESCRIPTION
<b>Frontend</b>	Inserted CEPAS card
<b>Backend</b>	Backend payment enabled at e-services portal
<b>BusinessFunctionDisabled</b>	No Business function is enabled

## 6.9.7 OBUTextColor

VALUES	DESCRIPTION
<b>Default</b>	Same as White
<b>White</b>	Represents color of the text for traffic messages
<b>Red</b>	Represent color of the text for traffic messages
<b>Blue</b>	Represents color of the text for traffic messages on the OBU Display
<b>Yellow</b>	Represents color of the text for traffic messages on the OBU Display
<b>Green</b>	Represents color of the text for traffic messages on the OBU Display
<b>Black</b>	Represents color of the text for traffic messages on the OBU Display

## 6.9.8 OBUTextStyle

VALUES	DESCRIPTION
<b>Default</b>	Same as Normal
<b>Normal</b>	Represents font weight of the text in for traffic messages on the OBU Display
<b>Bold</b>	Represents font weight of the text in for traffic messages on the OBU Display

## 6.9.9 OBUTrafficDataPriority

VALUES	DESCRIPTION
<b>High</b>	It represents emergency/high priority Traffic messages
<b>Charging</b>	ERP 2.0/ERP1/EEP/EPS/REP/OPC/CPT charging related messages  *REP/OPC/CPT (Reserved for future)
<b>Medium</b>	Messages related Information and driver operations
<b>Low</b>	Low priority Traffic messages

## 6.9.10 OBUTrafficMessageType

VALUES	DESCRIPTION
<b>UnplannedIncidentsType1</b>	This type has seven texts and one image. The examples of it shown in <b>Template1</b> .
<b>UnplannedIncidentsType11</b>	This type has seven texts and one image. The examples of it shown in <b>Template2</b> .
<b>FreeText5Line</b>	This type has five texts. The examples of it shown in <b>Template3</b> .
<b>FreeText6Line</b>	This type has six texts. The examples of it shown in <b>Template4</b> .
<b>FreeText7Line</b>	This type has seven texts. The examples of it shown in <b>Template5</b> .
<b>FreeText8Line</b>	This type has eight texts. The examples of it shown in <b>Template6</b> .
<b>PlannedIncidents</b>	This type has seven texts and one image. The examples of it shown in <b>Template8</b> .
<b>TravelTime</b>	This type has travel info for destinations. The examples of it shown in <b>Template10</b> .
<b>ParkingGuidance</b>	This type has parking info. The examples of it shown in <b>Template11</b> .
<b>PreloadedData</b>	This type has some texts and one image. The examples of it shown in <b>Template12</b> .
<b>ImageDisplay</b>	This type has one image. The examples of it shown in <b>Template13</b> .

### 6.9.11 BluetoothState (iOS)

It represents the current state of a CBManager.

VALUES	DESCRIPTION
<b>unknown</b>	State unknown, update imminent
<b>resetting</b>	The connection with the system service was momentarily lost, update imminent.
<b>unsupported</b>	The platform does not support the Bluetooth Low Energy role.
<b>unauthorized</b>	The application is not authorized to use the Bluetooth Low Energy role.
<b>poweredOff</b>	Bluetooth is currently powered off.
<b>poweredOn</b>	Bluetooth is currently powered on and available to use.

### 6.9.12 BluetoothState (Android)

This enum represents the states of 'ACTION\_STATE\_CHANGED' broadcast intent, which the system broadcasts whenever the Bluetooth state changes.

VALUES	DESCRIPTION
<b>STATE_TURNING_ON</b>	Indicates the local Bluetooth adapter is turning on.
<b>STATE_ON</b>	Indicates the local Bluetooth adapter is on, and ready for use.
<b>STATE_TURNING_OFF</b>	Indicates the local Bluetooth adapter is turning off.
<b>STATE_OFF</b>	Indicates the local Bluetooth adapter is off.

### 6.9.13 OBUErrorCode

VALUES	ERRORCODE	DESCRIPTION
<b>INTERNET_NOT_AVAILABLE</b>	1000	When there is no internet connectivity on the phone.
<b>BLUETOOTH_DISABLED</b>	1001	When the Bluetooth on the phone is disabled or turned off.
<b>DEVELOPER_UNAUTHORISED</b>	1003	When an SDK Account key issued to the developer is blocked.
<b>APPLICATION_UNAUTHORISED</b>	1004	When a specific application using an SDK Account issued to a developer is blocked.
<b>DEVELOPER_DEACTIVATED</b>	1005	In case if a developer is not allowed to use the SDK account key issued to them after registration at DataMall. You may contact customer support if this error persists.
<b>SEARCH_ERROR (for Android)</b>	1006	This is a specific error for Android related to companion device pairing scan. This is an internal error of that.
<b>OBU_NOT_PAIED</b>	2000	When the SDK tries to make a connection to the OBU which has not paired with the phone. This is specifically returned for the connect method via vehicle number.
<b>CONNECTION_FAILED</b>	2001	This is a generic error that occurs when connection with the OBU fails.
<b>DEVICE_DATA_ACCESS_ERROR</b>	2002	This represents an internal data error in the SDK while getting data from the OBU. Please contact customer support if this persists.
<b>SDK_AUTHENTICATION_REQUIRED</b>	2003	Calling the SDK APIs without authentication, throws this error code. The developer must initialise the SDK before calling the SDK methods.

<b>OBU_DATA_ACCESS_PERMISSION_DENIED</b>	2004	If OBU data permission is not taken from the motorist. The developer must ensure that this permission has been taken from the motorist in their applications. This is to inform the motorist about the OBU data being used by third-party applications.
<b>CODE_SIGNATURE_FAIL</b>	2009	This is an internal SDK data error. If this error persists, please contact customer support.
<b>NO_PUBLIC_KEY_AVAILABLE</b>	2010	This is an internal SDK data error. If this error persists, please contact customer support.
<b>BAD_REQUEST</b>	400	HTTP error code
<b>INVALID_SDK_ACCOUNT_KEY</b>	401	If the SDK account key is not correct
<b>REQUEST_TIMEOUT</b>	408	HTTP error code
<b>SERVER_ERROR</b>	500	HTTP error code
<b>MOCK_SEQUENCE_ENDED (for Android)</b>	3000	Returned when the mock manager sequence of events is terminated. This code can be used to find out the end of sequence. This is only used for the Mock Manager.
<b>MAC_ADDRESS_AUTHENTICATION_FAILED (iOS)</b>	2006	<p>User is required to register their iPhone's MAC Address with e-Services. If MAC Address entered is not registered/incorrect and an iPhone tries to make the connection, this error is returned in that case.</p> <p>Upon receiving this error, you can inform the user to enter the MAC Address next time when the connection request is initiated.</p> <p>The MAC Address User Interface is presented by the SDK during the connection process. It is required only one time. Once entered by the user, the User Interface does not appear again.</p>
<b>TRANSPORT_ERROR (iOS)</b>	616	This error is returned if the internet is connected but data packets cannot be sent. Only applicable for iOS.



## 7.SDK TROUBLESHOOTING

Here are some common issues you may come across when using the SDK and how to solve them.

### **How do I know the OBU is ready to pair with my smartphone?**

After switching on the car, the OBU takes 1-2 mins to be ready to pair with your smartphone. The OBU Display shows a Bluetooth icon at the top right corner of the screen when it is ready to pair and connect. Absence of it can mean two things: either your smartphone MAC Address has not been registered or if it is registered, the OBU will receive it in a while and start showing the icon. After that the OBU is ready to pair and connect to your phone.

### **I am not able to pair & connect my phone to the OBU?**

There is one main reason this might be happening. Your phone is not registered with the e-services Portal. Without registration the pairing and connection will not take place. Please refer to section 2.3 explaining the registration process for iPhone and Android.

### **What to do with the error codes (OBUErrorCode) and how to fix it?**

OBUErrorCode explains all the possible errors that you may receive while development and testing if something went wrong. If some of the error persists, please write us with detailed steps you have done at [extol@lta.gov.sg](mailto:extol@lta.gov.sg).

## APPENDIX A: TDCID ICONS

