OmronConnectivityLibrary

A Guide to integrate OmronConnectivityLibrary Framework with Third Party Apps

V 5.2

Revision History

Issue Date	Version	Details
06/24/2016	1.0	• Initial OmronConnectivityLibrary Integration Details
07/06/2016	1.1	• Added details on Scanning and Connecting Omron Connected Devices • Updated flow when Pairing and Transferring data • Added section to capture Omron Connected Device behavior
07/12/2016	1.2	• Updated Library and release version
08/02/2016	1.3	• Added details for TruRead Mode of Devices and Averaging Logic • Updated details to retrieve device model details like no of users, model name, and reference identifier.
12/08/2016	1.4	• Updated supported Omron device to include BP7000 (EVOLV) - III (2) • Updated section V (5) for connecting device • Added details on section V (6) about user type • Update section V (8) for disconnecting device • Device Information details – Section V (9) • Updated Error Handling and Connectivity Scenarios – Section VII (2) and (3) • Added new screenshot for BP7000 pairing tutorial (EVOLV) in appendix Section VIII
12/12/2016	1.5	• Updated to add details on Omron BLE protocol for BP7000 • Update section IV (6) to add more details • Updated section V (5) for connecting device for BLP Profile • Device Information details – Section V (9) for BLP Profile • Updated Error Handling and Connectivity Scenarios – Section VII (2) and (3) and (4) for BLP Profile
10/01/2017	2.0	• New Version of Library with improved connectivity support • Updated supported devices and development environment requirements • More details added for Retrieve Configuration V (2) • Additional support for Settings Configuration V (3) • Updated Stop Scan featurre V (4) • More information added for Section V (9) and updated specification for V (9) (b) • Added details about Error code Section VIII • Removed specification for BLP device support
01/17/2018	2.1	• Updated VI (1) to include a new helper function to retrieve device configuration
02/09/2018	2.2	• Updated V (2) to include new notification functionality to share device configuration status from library
02/12/2018	2.3	• Updated V (6) for new feature of pause and resume data transfer
02/21/2018	2.4	• Updated configuration file BP786/CAN and BP761/CAN
04/05/2018	2.5	• Updated V (3) to add details on User Hash Id property
06/01/2018	2.6	• Added support for Omron devices BP6000 / BP6001 (HeartVue) - III (2) • Update section V (1) to capture Partner authentication error code

		• Added details for setting configuration under section V (2) • Added details of Activity device under Section V (6) and Section V (7) • New function to update Omron Activity device – Section V (9) • Updated Appendix VIII (2) for error code for encryption failure • Added new Appendix VIII (3) for sample output data
07/01/2018	2.7	• Added support for Omron device BP7900
09/04/2018	2.8	• Add Google Analytics Integration to connectivity
09/11/2018	2.9	• Add support for BP300
09/14/2018	3.0	• Update Google Analytic to track additional details
12/06/2018	3.1	• Update device model name for BP7900
01/07/2019	3.2	• Add support for Omron device M700 Intelli IT
01/21/2019	3.3	• Add support for Omron device BP8000 (HeartGuide)
02/20/2019	3.4	• Add support for Omron device BP6350, BP4350, BP7250, BP7450 • Add new keys available in blood pressure in section V (7) • Update section V (10) (a) to capture details on iBeacon
04/08/2019	3.5	• Add support for Omron device HBF-222T_Z • Added new Appendix IV (5) for instructions to add assets bundle • Updated section V (3) (d) (I) to capture new keys under personal settings • Updated section V (3) (d) to capture sleep for Activity • Added new methods in Appendix V (5) for connecting Omron connected devices • Added new methods in Appendix V (9) under update device settings • Added new Appendix V (10) (b) for device settings • Added new Appendix VIII (3) (c) for device settings in sample output data from library
05/01/2019	3.6	• Add support for Omron device EVOLV, RS7 Intelli IT, VIVA
06/04/2019	3.7	• Add support for Omron device BP5250, BP5350, BP5450
11/04/2019	3.8	• Add support for Omron device BP7250CAN, BP7350CAN, BP7450CAN
11/10/2019	3.9	• Add support for RS3 Intelli IT, HeartGuide, MIT5s, M4 Intelli IT, M7 Intelli IT 2nd Gen
03/03/2020	4.0	• Update to support Xcode 11 • Remove support for Google Analytics
03/31/2020	4.1	• Improvements in connectivity
06/18/2020	4.2	• Update image asset catalog • Improvements in connectivity • Support unique identifier for library • Add support for HEM-7280T-AP, HEM-7600T-AP3, HBF-222T-APW, HEM-6232T-AP, HEM-7361T-AP
08/22/2020	4.3	• Update asset catalog • Additional feature support for connectivity pairing and data transfer

09/15/2020	4.4	• Support for NightView (HEM-9601T) • Improvements to connectivity functionalities
01/05/2021	4.5	• Update device support
03/16/2021	4.6	• Updated iOS Deployment Target to iOS 10.0 • Update library to support Xcode 12 • Update to asset catalog for device images • Add weight only measurement configuration for body composition monitor • Update device support
05/05/2021	4.7	• Improvements to connectivity functionalities • Update to asset catalog for device images • Add body composition features for guest user for BCM devices • Update device support
07/15/2021	4.8	• Improvements to connectivity functionalities • Update for Blood pressure measurement mode flag
09/10/2021	4.9	• Add support for HWZ-1000T-E, MC-280B, HPO-300T, HN-300T2, HEM-7530T-E3 • Improvements to connectivity functionalities
10/15/2021	5.0	• Add support for SC-150, M2 Intelli IT, X2 Smart, M300 Intelli IT • Improvements to connectivity functionalities
11/05/2021	5.1	• M2 Intelli IT support fix • Improvements to connectivity functionalities
16/02/2022	5.2	• Improvements to connectivity functionalities

Table of Contents

Getting Started

I. Introduction

The OmronConnectivityLibrary allows Partner iOS applications to interact with Omron Connected Devices. This allows partners to retrieve vital data from Omron Connected devices to the iOS device. Partners can use the data to build feature-rich user experience. This document will guide you to add OmronConnectivityLibrary to an iOS project, as well as introducing the Library's API and how to communicate with Omron Connected Devices.

II. Version

OmronConnectivityLibrary Release Version: 3.0.23

III. Prerequisites

1. Development Environment

- Using Xcode 11.3.1 or higher.
- Targeting iOS 10.0 or higher.
- Valid Apple Developer Account and Provisioning Profile

2. Supported Smartphone

For a list of compatible smartphones and OMRON devices please visit the following links:

- US/Canada devices: https://omronhealthcare.com/service-and-support/connected-health/connected-device-compatibility/
- Other devices: https://www.omronconnect.com/emea/en_gb/devices/

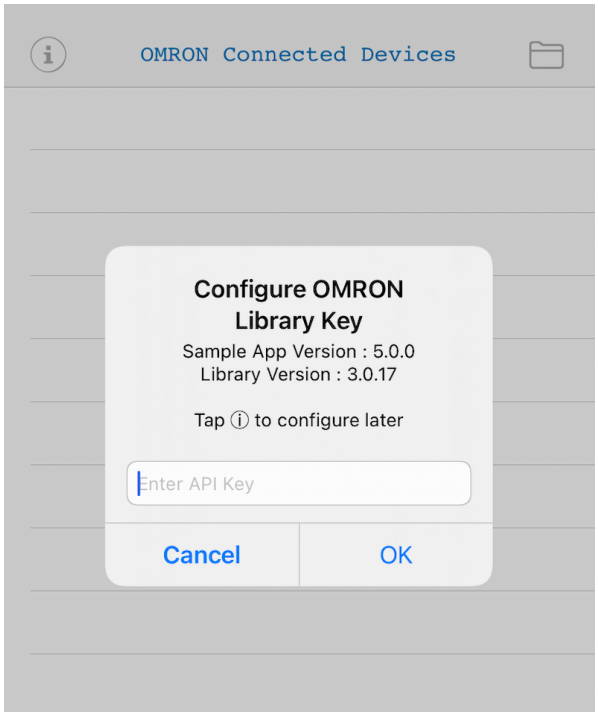
IV. Running the Sample Application

1. Install the sample app

Open OmronLibrarySample Xcode project available with SDK and generate build to install in device.

2. Configuring Partner API key

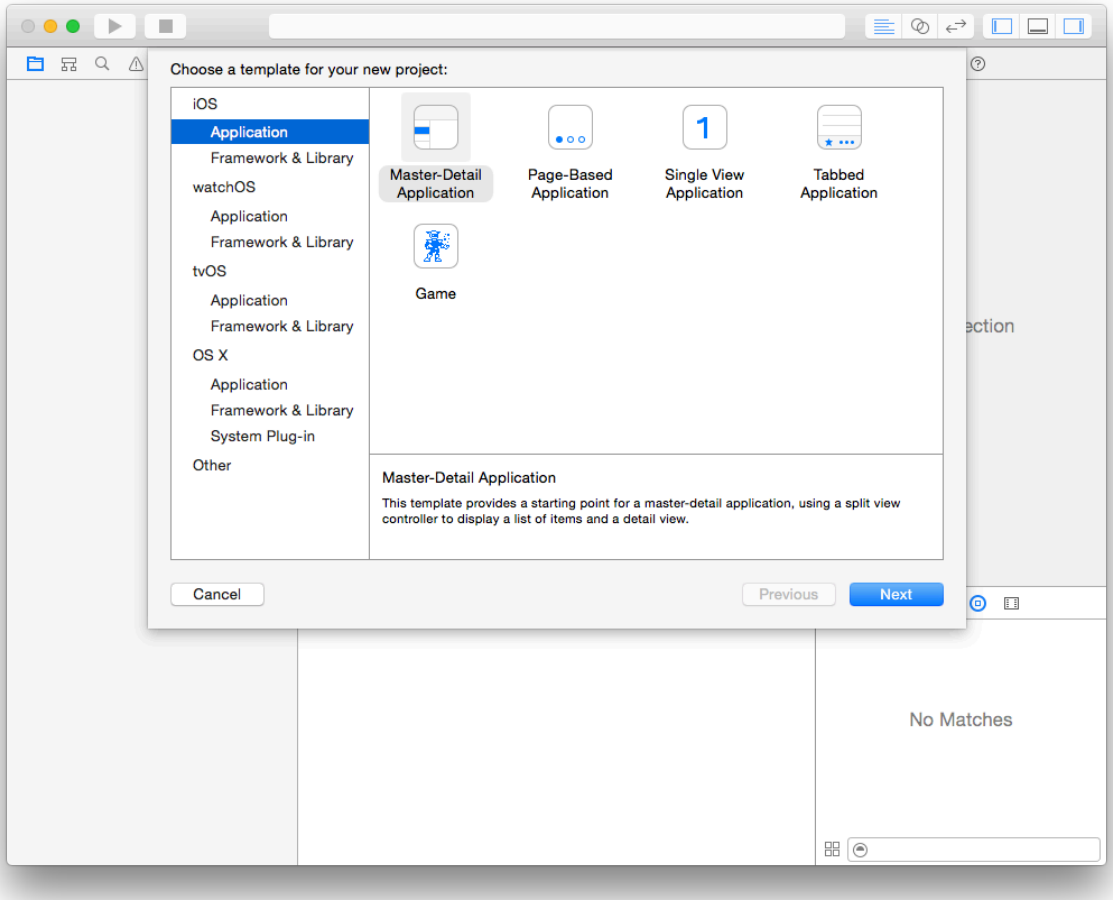
Once build is installed, open the application and configure the Library key available and follow instructions in screen. Partner's API key is used to validate the authenticity of Partner.



V. Configuring Project for OmronConnectivityLibrary

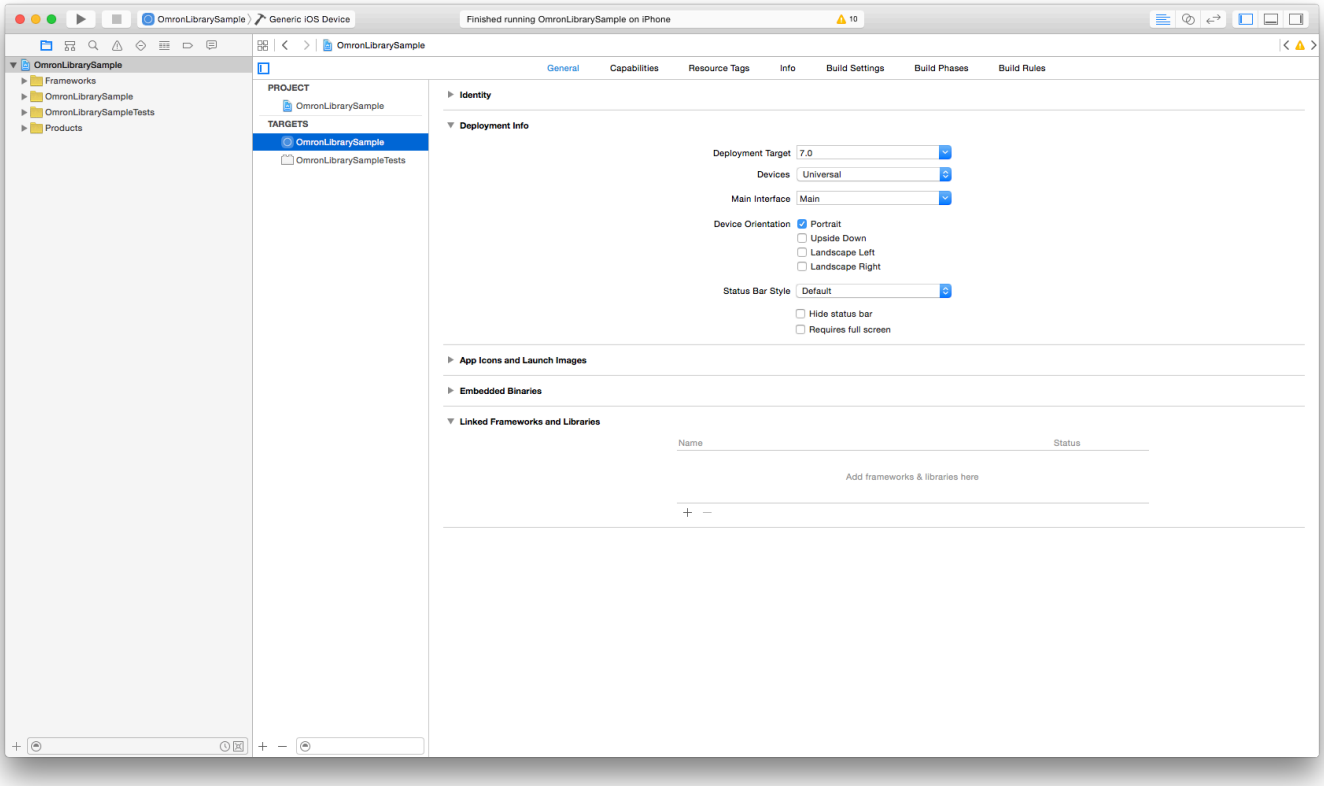
1. Create new Xcode Project

In Xcode application select File > New > Project and create a new Application. Select the desired type of Application under iOS and create the Xcode Project.

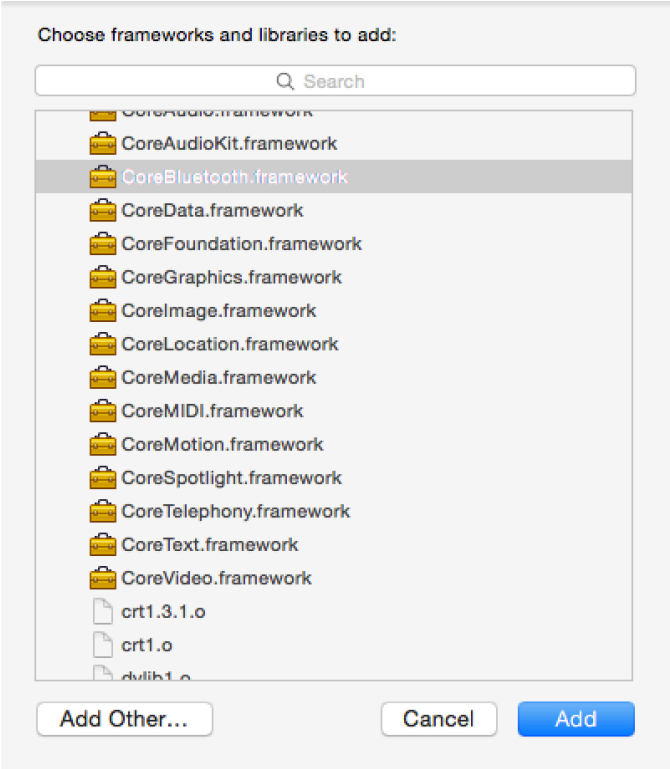


2. Add CoreBluetooth Framework

CoreBluetooth.framework of iOS is required for Bluetooth connectivity with Omron Connected devices. Add this framework under Target > General > Linked Frameworks and Libraries



Now click on “+” and select CoreBluetooth.framework from the list to add.

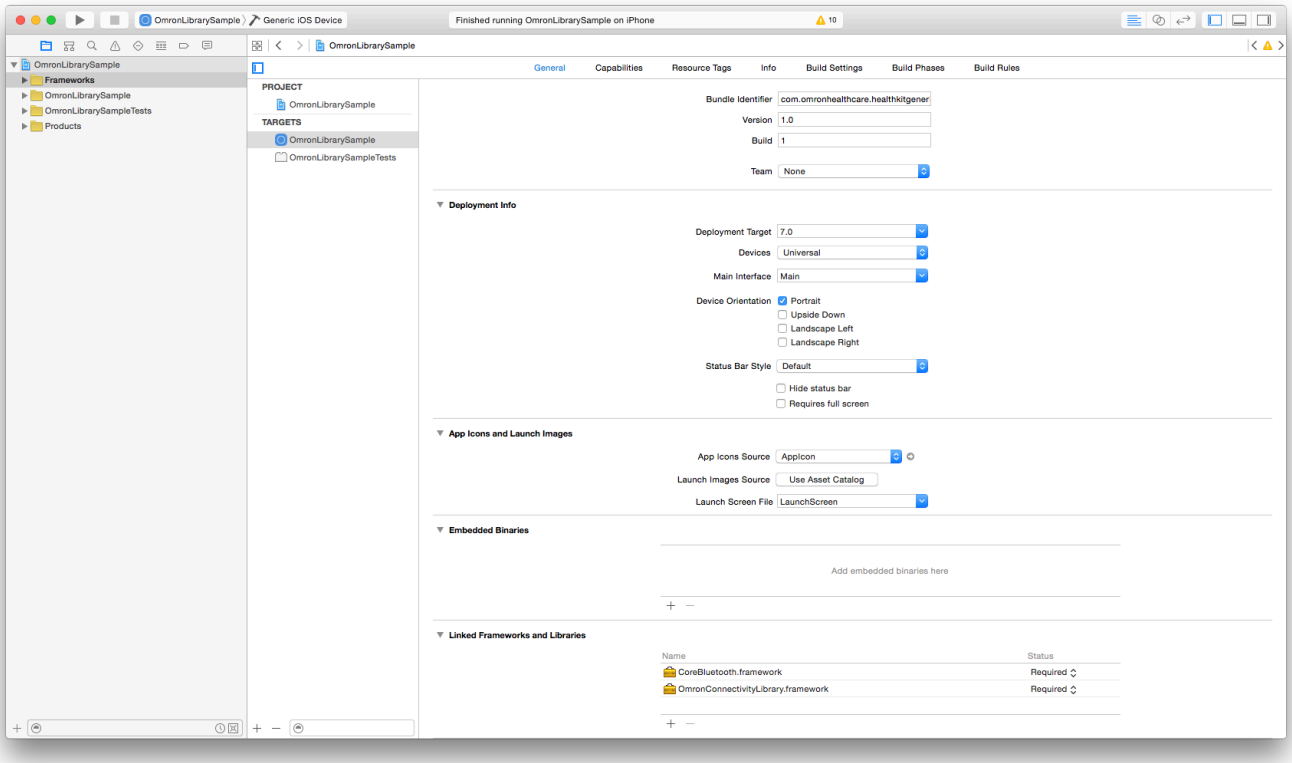


Note: Add CoreBluetooth usage descriptions to project following Apple’s Privacy permissions .

3. Add OmronConnectivityLibrary Framework

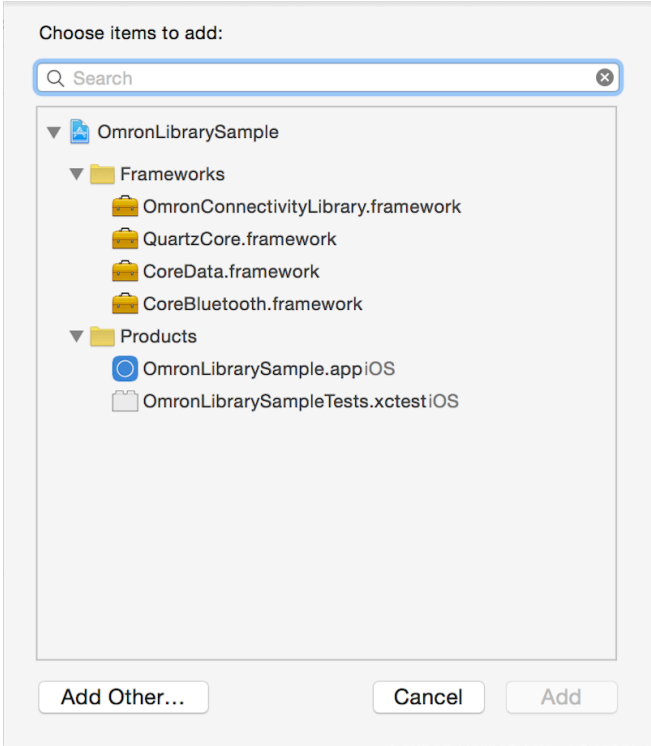
OmronConnectivityLibrary.framework is required to connect to Omron Connected Devices. Right click project name and choose "Add Files to… " Option to add this framework to Xcode project.

Select "Copy items if needed" when adding the framework to bundle the framework in Partner application. Once completed confirm that target has the framework included



4. Embed OmronConnectivityLibrary Framework

Add the OmronConnectivityLibrary.framework file as embed framework under "Embedded Binaries" in Target > General > Embedded Binaries

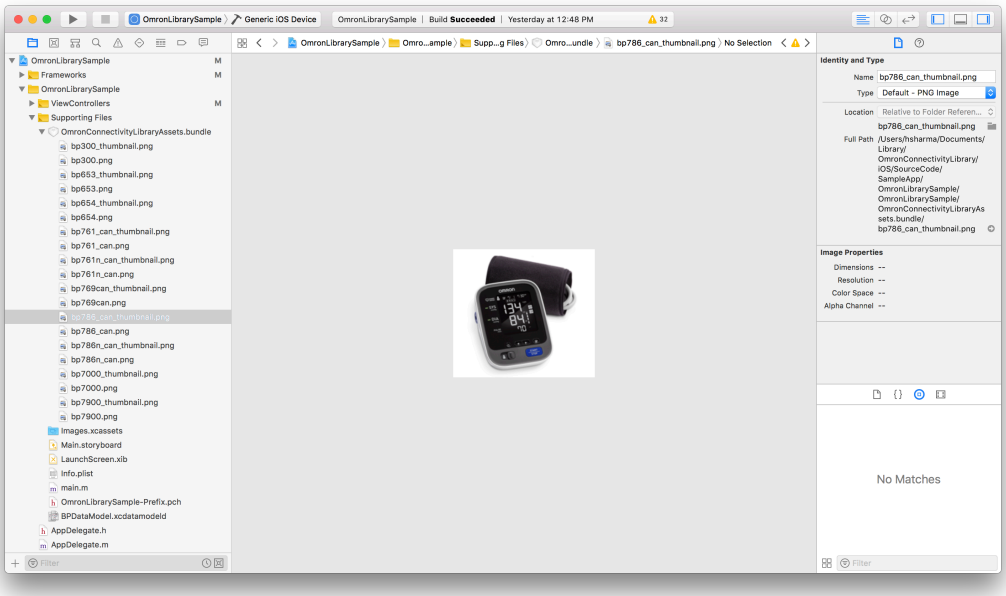


Reference:
https://developer.apple.com/library/content/technotes/tn2435/_index.html

5. Add OmronConnectivityLibraryAssets Bundle (optional)

`OmronConnectivityLibraryAssets.bundle` is required to get the image and thumbnails of Omron Connected Devices. Right click project name and choose "Add Files to… " option to add this asset bundle to Xcode project.

Please select "Copy items if needed" when adding the asset bundle to bundle the same in Partner application. Once completed confirm that target has the bundle included. Using bundle will increase the size of your application package.



6. Verify Targets

Verify that the OmronConnectivityLibrary.framework Target Membership checkbox is set. Also verify that your Deployment Target is greater than or equal to deployment target of library. Deployment target of OmronConnectivityLibrary framework is 8.2.

VI. OmronConnectivityLibrary Framework Integration

1. Initializing OmronConnectivityLibrary

Include OmronConnectivityLibrary.h header file to use features of Omron Connectivity Library. All interaction with the Library is done through the OmronPeripheralManager Class. This class must be initialized during app startup with the Partner's API Key. Partner's API Key is used to validate the authenticity of Partner.

```
#import "AppDelegate.h"
#import <OmronConnectivityLibrary/OmronConnectivityLibrary.h>

@interface AppDelegate ()

@end

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {

        [[OmronPeripheralManager sharedManager] setAPIKey:@"PARTNER_API_KEY" options:nil];

   return YES;
}
```

This is a mandatory step to start using Omron Connectivity Library. Typically this is done within the AppDelegate implementation file (.m) in method application:didFinishLaunchingWithOptions:

Partner authentication is verified based on below status code when using Library functionalities.

CODE	DESCRIPTION
OMRONConfigurationPartnerAuthenticationError	Partner is not authorized

2. Retrieve Configurations

Omron Device configurations need to be retrieved to identify different Omron Connected Devices available for connection. These configurations are used for discovering, connecting and transferring vital data from Omron Connected Devices. To retrieve details from OmronConnectivityLibrary, partner application need to invoke below function.

```
[[OmronPeripheralManager sharedManager] retrieveManagerConfiguration]
```

This function returns a dictionary of configuration, in which Omron Connected Devices can be retrieved using key OMRONBLEConfigDeviceKey. This is a list of Omron Connected devices. Each Omron device item in list has below properties.

KEY	DESCRIPTION
OMRONBLEConfigDeviceModelNameKey	Device Sales Name
OMRONBLEConfigDeviceModelSeriesKey	Device Series Type
OMRONBLEConfigDeviceUsersKey	No of users available in memory
OMRONBLEConfigDeviceCategory	Device Category
OMRONBLEConfigDeviceGroupIDKey	Device Category Type
OMRONBLEConfigDeviceGroupIncludedGroupIDKey	Device Model Type
OMRONBLEConfigDeviceIdentifierKey	Device Identifier / Product Code
OMRONBLEConfigDeviceProtocolKey	Device Communication Protocol Standard
OMRONBLEConfigDeviceImageKey	Device main image path
OMRONBLEConfigDeviceThumbnailKey	Device thumbnail image path

Device image and thumbnail image can be used by `OMRONBLEConfigDeviceImageKey` and `OMRONBLEConfigDeviceThumbnailKey`. Asset bundle is needed to be added first into the project before using these keys to show device images and thumbnails. See section V 5.

```
// Get Devices List from Configuration File in Framework
NSDictionary *configDictionary = [[NSDictionary alloc]
initWithDictionary:[[OmronPeripheralManager sharedManager]
retrieveManagerConfiguration]];

deviceList = [NSMutableArray arrayWithArray:[configDictionary
objectForKey:OMRONBLEConfigDeviceKey]];

for (NSDictionary *device in deviceList) {
        UIImage *deviceImage = [UIImage imageNamed:[device
valueForKey:OMRONBLEConfigDeviceImageKey]];
        UIImage *deviceThumbnail = [UIImage imageNamed:[device
valueForKey:OMRONBLEConfigDeviceThumbnailKey]];
}
```

Framework posts notification to partner application when configuration is ready for use. Partner application is required to listen to these before using any connectivity related functions.

Different status codes posted by framework are:

CODE	DESCRIPTION
OMRONConfigurationFileSuccess	Configuration setup success
OMRONConfigurationFileError	Configuration setup failure
OMRONConfigurationFileUpdateError	Configuration upgrade failure

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(configAvailabilityNotification:)
                                             name:OMRONBLEConfigDeviceAvailabilityNotification
                                           object:nil];

- (void)configAvailabilityNotification:(NSNotification *)aNotification {

   OMRONConfigurationStatus configFileStatus = (OMRONConfigurationStatus)[aNotification.object
unsignedIntegerValue] ;

   if(configFileStatus == OMRONConfigurationFileSuccess) {
          // Configuration extract success
   }else if(configFileStatus == OMRONConfigurationFileError) {
         // Configuration extract failure – App
   }else if(configFileStatus == OMRONConfigurationFileUpdateError) {
         // Configuration update failure
   }
}
```

3. Setting Configurations

Partner application is provided with option to set configurations to Omron devices. Below are the available configurations.

(a) Timeout Interval (optional) – Determines the scan timeout interval when searching for Omron Connected Devices. The default timeout is 60 seconds.

(b) Device Filters (optional) – Helps partner application to filter the devices to particular Omron device models when discovering Omron Connected Devices. Details are retrieved from configurations fetched in step VI (2)

(c) User Hash Id (**mandatory**) – This is for authenticating connection between Omron devices and application during Pair and Transfer. This is preferably the user's login email address, which will be used in mobile app for authentication purpose. Application need to ensure that this property is configured properly and is identical when using Pair and Transfer functionalities. Also it should remain consistent between app upgrades. If different input is provided, device will fail to connect or transfer due to encryption issue. See Appendix.

(d) Library identifier – Returns a unique identifier for the library. This remains constant per app install.

(e) Read historic data – Provide capability to read all readings from device for selected user. Application need to update this flag accordingly in subsequent data transfer if used once. This is not supported for some device category types.
Disclaimer: Guest data from device will also get transferred by enabling this. Use only if required.

(f) Read iBeacon details in data transfer – Fetches iBeacon details from device during data transfer. Pairing always returns iBeacon details.
Disclaimer: Use only if required. Bluetooth performance impact if enabled.

(g) Device Settings - This is used to set configuration in device.

KEY	DESCRIPTION	DEVICE TYPE
OMRONDevicePersonalSettingsKey	Personal settings for the user	Activity and Body Composition
OMRONDeviceTimeSettingsKey	Device Time format settings	Activity
OMRONDeviceDateSettingsKey	Device Date format settings	Activity
OMRONDeviceDistanceSettingsKey	Device Distance format settings	Activity

OMRONDeviceSleepSettingsKey	Sleep settings values	Activity
OMRONDeviceAlarmSettingsKey	Alarm settings values	Activity
OMRONDeviceWeightSettingsKey	Weight settings values	Body Composition

I. Personal settings (mandatory for mentioned device categories):
For activity device - Height, weight and stride
For weighing scale - Height, date of birth, gender and DCI

Personal settings `OMRONDevicePersonalSettingsKey` will have the following details

KEY	DESCRIPTION	UNIT	DEVICES
OMRONDevicePersonalSettingsUserHeightKey	Height of the user	cm	Activity, Body Composition
OMRONDevicePersonalSettingsUserWeightKey	Weight of the user	kg	Activity
OMRONDevicePersonalSettingsUserStrideKey	Stride of the user	cm	Activity
OMRONDevicePersonalSettingsUserDateOfBirthKey	Date of birth of the user	yyyymmdd	Body Composition
OMRONDevicePersonalSettingsUserGenderKey	Gender of the user	0 (female) 1 (male)	Body Composition
OMRONDevicePersonalSettingsWeightKey	Other weight Settings	-	Body Composition
OMRONDevicePersonalSettingsBloodPressureKey	Other blood Pressure Settings	-	Blood Pressure
OMRONDevicePersonalSettingsTargetStepsKey	Target number of steps	count	Activity
OMRONDevicePersonalSettingsTargetSleepKey	Target sleep	minutes	Activity

Calculation of Height/Weight/Stride:
If “a” is the value of Height (cm) or Stride (cm) or Weight (kg), value that need to be passed to Connectivity configuration settings will be:

Configuration Value = roundf(a X 100)

It is important that the application need to pass the proper data to library based on the unit defined for each data type – height/weight/stride. If accurate value is not passed, the device will calculate invalid steps, calories, and so on.

DCI and Calculation of DCI:
DCI is an increment value that requires to be sent to the device while pairing or updating the device settings. App will update the device's personal settings like date of birth, gender, height and weight unit based on the DCI received.
Omron device will accept the settings sent from the App only if DCI value sent from device is greater than the DCI value currently saved in Omron device memory.

For first time pairing, the value to this key will be below
Weight: `OMRONDevicePersonalSettingsWeightDCINotAvailable`
Blood pressure:
`OMRONDevicePersonalSettingsBloodPressureDCINotAvailable`

Blood pressure settings:
`OMRONDevicePersonalSettingsBloodPressureKey` has following details.

KEY	DESCRIPTION	UNIT
OMRONDevicePersonalSettingsBloodPressureDCIKey	DCI Value for Blood pressure	Int
OMRONDevicePersonalSettingsBloodPressureTruRead EnableKey	TruRead Enable Key	*Refer below
OMRONDevicePersonalSettingsBloodPressureTruRead IntervalKey	TruRead Interval Key	*Refer below

`OMRONDevicePersonalSettingsBloodPressureTruReadEnableKey` and `OMRONDevicePersonalSettingsBloodPressureTruReadIntervalKey` can have the following possible values. These features are available for specific blood pressure devices only. Refer instruction manual of device to learn more about the device feature.

KEY	VALUES
OMRONDevicePersonalSettingsBloodPressureTruReadEnableKey	OMRONDevicePersonalSettingsBloodPressureTruReadOff OMRONDevicePersonalSettingsBloodPressureTruReadOn
OMRONDevicePersonalSettingsBloodPressureTruReadIntervalKey	OMRONDevicePersonalSettingsBloodPressureTruReadInterval15 OMRONDevicePersonalSettingsBloodPressureTruReadInterval30 OMRONDevicePersonalSettingsBloodPressureTruReadInterval60 OMRONDevicePersonalSettingsBloodPressureTruReadInterval120

Weight settings:
`OMRONDevicePersonalSettingsWeightKey` has following details.

KEY	DESCRIPTION	UNIT
OMRONDevicePersonalSettingsWeightDCIKey	DCI Value for Weight	Int

Please refer the below table for possible use cases for DCI for blood pressure and weight.

DCI Device	DCI App	New DCI (device)	Result in Device
0	-1 (DCI not available)	1	Settings updated
1	1	1	Settings not updated
1	2	2	Settings updated
3	2	3	Settings not updated

II. Device Time settings (activity device only - optional)

Device time settings `OMRONDeviceTimeSettingsKey` will have the following details

KEY	DESCRIPTION	VALUES
OMRONDeviceTimeSettingsFormatKey	Key to specify the device time format.	OMRONDeviceTime24Hour - 24 Hour OMRONDeviceTime12Hour - 12 Hour

III. Device Date settings (activity device only - optional)

Device date settings `OMRONDeviceDateSettingsKey` will have the following details

KEY	DESCRIPTION	VALUES
OMRONDeviceDateSettingsFormatKey	Key to specify the device date format.	OMRONDeviceDateFormatMonthDay - MMDD OMRONDeviceDateFormatDayMonth - DDMM

IV. Device Distance settings (activity device only - optional)

Device distance settings `OMRONDeviceDistanceSettingsKey` will have the following details

KEY	DESCRIPTION	VALUES
OMRONDeviceDistanceSettingsUnitKey	Key to specify the device distance format.	OMRONDeviceDistanceUnitKilometer - Kilometer OMRONDeviceDistanceUnitMile - Miles

V. Device Sleep settings (activity device only - optional)

Device sleep settings `OMRONDeviceSleepSettingsKey` will have the following details

KEY	DESCRIPTION	VALUES
OMRONDeviceSleepSettingsAutomaticKey	Auto sleep setting	OMRONDeviceSleepAutomaticOn / OMRONDeviceSleepAutomaticOff
OMRONDeviceSleepSettingsAutomaticStartTimeKey	Start time	0~23
OMRONDeviceSleepSettingsAutomaticStopTimeKey	Stop time	0~23

VI. Device Alarm settings (activity device only - optional)

Alarm Settings `OMRONDeviceAlarmSettingsKey` will be a list of alarm item. A maximum of 5 alarms can be set on an activity device. Each Item will have the following values

KEY	DESCRIPTION
OMRONDeviceAlarmSettingsDaysKey	Alarm days
OMRONDeviceAlarmSettingsTimeKey	Alarm time
OMRONDeviceAlarmSettingsTypeKey	Alarm Type

Alarm time `OMRONDeviceAlarmSettingsTimeKey` will have the following key-values

KEY	DESCRIPTION
OMRONDeviceAlarmSettingsHourKey	Hour (Always in 24-hour format)
OMRONDeviceAlarmSettingsMinuteKey	Minute

Alarm days `OMRONDeviceAlarmSettingsDaysKey` will have the following key-values

KEY	DESCRIPTION	VALUE
OMRONDeviceAlarmSettingsDaySundayKey	Set Alarm for Sunday	OMRONDeviceAlarmStatusOn – On OMRONDeviceAlarmStatusOn - Off
OMRONDeviceAlarmSettingsDayMondayKey	Set Alarm for Monday	OMRONDeviceAlarmStatusOn – On OMRONDeviceAlarmStatusOn - Off
OMRONDeviceAlarmSettingsDayTuesdayKey	Set Alarm for Tuesday	OMRONDeviceAlarmStatusOn – On OMRONDeviceAlarmStatusOn - Off
OMRONDeviceAlarmSettingsDayWednesdayKey	Set Alarm for Wednesday	OMRONDeviceAlarmStatusOn – On OMRONDeviceAlarmStatusOn - Off
OMRONDeviceAlarmSettingsDayThursdayKey	Set Alarm for Thursday	OMRONDeviceAlarmStatusOn – On OMRONDeviceAlarmStatusOn - Off
OMRONDeviceAlarmSettingsDayFridayKey	Set Alarm for Friday	OMRONDeviceAlarmStatusOn – On OMRONDeviceAlarmStatusOn - Off
OMRONDeviceAlarmSettingsDaySaturdayKey	Set Alarm for Saturday	OMRONDeviceAlarmStatusOn – On OMRONDeviceAlarmStatusOn - Off

Alarm type `OMRONDeviceAlarmSettingsTypeKey` will have the following key-values

KEY	DESCRIPTION
OMRONDeviceAlarmTypeNormal	Normal alarm
OMRONDeviceAlarmTypeMeasure	Blood pressure measurement alarm
OMRONDeviceAlarmTypeMedication	Medication alarm

VII. Device Weight settings (Body Composition only - optional)

Device weight settings `OMRONDeviceWeightSettingsKey` will have the following details

KEY	DESCRIPTION	VALUE
OMRONDeviceWeightSettingsUnitKey	Key to specify the device weight unit. Supported for BCM and Weight scale device	OMRONDeviceWeightUnitKg – Kilogram OMRONDeviceWeightUnitLbs – Lbs OMRONDeviceWeightUnitSt – St
OMRONDeviceWeightSettingsOneTimeMeasurementKey	Key to specify if device need to be registered for guest mode	OMRONDeviceWeightOneTimeMeasurementOn – Guest mode OMRONDeviceWeightOneTimeMeasurementOff – Normal mode
OMRONDeviceWeightSettingsWeightOnlyKey	Key to specify if device need to be registered for weight only feature	OMRONDeviceWeightOnlyOn– Weight only feature OMRONDeviceWeightOnlyOff– All BCM features. This is default.

Setting weight unit will also automatically set the height unit in the device. Check below table for details about what will be the height unit when we change the weight unit to any possible value from the above table.

WEIGHT UNIT	HEIGHT UNIT
OMRONDeviceWeightUnitKg – Kilogram	Cms (Centimeters)
OMRONDeviceWeightUnitLbs – Lbs	Ft - Inch (Feet - Inch)
OMRONDeviceWeightUnitSt – St	Ft - Inch (Feet - Inch)

One Time Key	Weight Only Key	OUTPUT
Yes	YES	Guest mode + Weight only
YES	NO	Guest mode + BCM data
NO	YES	Regular mode with BCM data
NO	NO	Regular mode with BCM data

When BCM device is in “one time” mode, then only single reading can and transferred at a time. Also based on “weight only” mode, the data output from device changes based on input shared.

When “one time” mode is ON and “weight only” mode is OFF, then personal information required for body composition device need to send in order to calculate and display body composition data.

When “one time” mode is ON and “weight only” mode is ON, except weight, all other data returned by the library is invalid. After configuring the device in “weight only” mode during pairing, user can take weight reading. Once weight measurement is completed, device goes into transfer mode automatically and application can transfer data from device. Device will go into “Err” mode if no readings are available and data transfer is performed. Transferring historic data is not supported with this feature.

Omron Connectivity Library provides mechanism to set these above configurations.

Setting Configuration – This method lets application set configuration while Pairing or Data Transfer with Omron devices.

```
// Set Configuration to New Configuration
[[OmronPeripheralManager sharedManager] setConfiguration:config];
```

Library provides error code if settings are missing or invalid while pairing or transferring.

CODE	DESCRIPTION
OMRONConfigurationMissingParameterError	Configuration required for pair/transfer missing
OMRONConfigurationUserMismatchError	Incompatible user number for selected device
OMRONConfigurationUserHashMissingParameterError	User Hash is not set for configuration/encryption
OMRONConfigurationUserMismatchError	User number mismatch

4. Details of Setting Configurations with code example

Configure settings for connectivity library:

```
// Filter List
NSMutableArray *filterDevices = [[NSMutableArray alloc] init];

// Device Model available in retrieveManagerConfiguration
[filterDevices addObject:_modelDetails];

// Peripheral Configuration
OmronPeripheralManagerConfig *peripheralConfig = [[OmronPeripheralManager sharedManager]
getConfiguration];
peripheralConfig.deviceFilters = filterDevices; // Device Filter List
peripheralConfig.timeoutInterval = 15; // Seconds
peripheralConfig.userHashId = @"<logged in user email>"; // Set User email
```

Model details can be retrieved from the `OmronPeriperhalManager` using `retrieveManagerConfiguration` and from this list can be retrieved using `OMRONBLEConfigDeviceKey.` This list will have the details of all device models.

Retrieve Omron device model details from `OmronPeriperhalManager`

```
// Get Devices List from Configuration File in Framework
NSDictionary *configDictionary = [[NSDictionary alloc]
initWithDictionary:[[OmronPeripheralManager sharedManager]
retrieveManagerConfiguration]];

deviceList = [NSMutableArray arrayWithArray:[configDictionary
objectForKey:OMRONBLEConfigDeviceKey]];

_modelDetails = [NSMutableDictionary dictionaryWithDictionary:[deviceList
objectAtIndex:indexOfSelectedDevice]];
```

Configuration for Blood Pressure Device:

```
// Blood Pressure Settings (Personal settings)
NSDictionary *bloodPressurePersonalSettings = @{
OMRONDevicePersonalSettingsBloodPressureDCIKey:
@(OMRONDevicePersonalSettingsBloodPressureDCINotAvailable),

OMRONDevicePersonalSettingsBloodPressureTruReadEnableKey :
@(OMRONDevicePersonalSettingsBloodPressureTruReadOn),

OMRONDevicePersonalSettingsBloodPressureTruReadIntervalKey :
@(OMRONDevicePersonalSettingsBloodPressureTruReadInterval30)
                                                        };
NSDictionary *settings = @{ OMRONDevicePersonalSettingsBloodPressureKey :
bloodPressurePersonalSettings };

NSMutableDictionary *personalSettings = [[NSMutableDictionary alloc] init];
[personalSettings setObject:settings forKey:OMRONDevicePersonalSettingsKey];

NSMutableArray *deviceSettigs = [[NSMutableArray alloc] init];
[deviceSettigs addObject:personalSettings];

// Set Device Settings
peripheralConfig.deviceSettings = deviceSettigs;

// Set Configuration to New Configuration
[[OmronPeripheralManager sharedManager] setConfiguration:peripheralConfig];
```

Configuration for Activity Devices:

a) Device and personal settings

```
NSDictionary *settings = @{
OMRONDevicePersonalSettingsUserHeightKey : <HEIGHT (cm)>,
OMRONDevicePersonalSettingsUserWeightKey : <WEIGHT (kg)>,
OMRONDevicePersonalSettingsUserStrideKey : <STRIDE (cm)>};

NSMutableDictionary *personalSettings = [[NSMutableDictionary alloc] init];
[personalSettings setObject:settings forKey:OMRONDevicePersonalSettingsKey];

NSMutableArray *deviceSettigs = [[NSMutableArray alloc] init]; [deviceSettigs
addObject:personalSettings]; peripheralConfig.deviceSettings = deviceSettigs;

// Set Device Settings
peripheralConfig.deviceSettings = deviceSettigs;

// Set Configuration to New Configuration
[[OmronPeripheralManager sharedManager] setConfiguration:peripheralConfig];
```

b) Alarm Settings, Time and Date Settings:

```
// Alarm1 Time
NSMutableDictionary *alarmTime1 = [[NSMutableDictionary alloc] init];
[alarmTime1 setValue:"" forKey:OMRONDeviceAlarmSettingsHourKey];
[alarmTime1 setValue:"" forKey:OMRONDeviceAlarmSettingsMinuteKey];

// Alarm1 Days (SUN-SAT)
// Enable – 1, Disable - 0
NSMutableDictionary *alarmDays1 = [[NSMutableDictionary alloc] init];
[alarmDays1 setObject: @"1" forKey: OMRONDeviceAlarmSettingsDaySundayKey];
[alarmDays1 setObject: @"0" forKey: OMRONDeviceAlarmSettingsDayMondayKey];
[alarmDays1 setObject: @"1" forKey: OMRONDeviceAlarmSettingsDayTuesdayKey];
[alarmDays1 setObject: @"1" forKey: OMRONDeviceAlarmSettingsDayWednesdayKey];
[alarmDays1 setObject: @"0" forKey: OMRONDeviceAlarmSettingsDayThursdayDay];
[alarmDays1 setObject: @"1" forKey: OMRONDeviceAlarmSettingsDayFridayDayKey];
[alarmDays1 setObject: @"1" forKey: OMRONDeviceAlarmSettingsDaySaturdayKey];

// Set Alarm and Time Settings
NSMutableDictionary *alarm1 = [[NSMutableDictionary alloc] init];
[alarm1 setObject: alarmTime1 forKey: OMRONDeviceAlarmSettings.Days];
[alarm1 setObject: alarmDays1 forKey: OMRONDeviceAlarmSettings.Time];

// Add Alarm1 to List. This list will be passed in above code snippet.
NSMutableArray *alarms = [[NSMutableArray alloc] init];
[alarms addObject: alarm1];

// Set Alarms options
NSMutableDictionary *alarmSettings = [[NSMutableDictionary alloc] init];
[alarmSettings setObject:alarms forKey:OMRONDeviceAlarmSettingsKey];

// Retrieve previous configuration of OmronPeripheralManagerConfig
OmronPeripheralManagerConfig *peripheralConfig = [[OmronPeripheralManager sharedManager]
getConfiguration];

// Time Format
NSDictionary *timeFormatSettings = @{ OMRONDeviceTimeSettingsFormatKey :
@(OMRONDeviceTimeFormat24Hour) };
NSMutableDictionary *timeSettings = [[NSMutableDictionary alloc] init];
[timeSettings setObject:timeFormatSettings forKey:OMRONDeviceTimeSettingsKey];

// Date Format
NSDictionary *dateFormatSettings = @{ OMRONDeviceDateSettingsFormatKey :
@(OMRONDeviceDateFormatDayMonth) };
NSMutableDictionary *dateSettings = [[NSMutableDictionary alloc] init];
[dateSettings setObject:dateFormatSettings forKey:OMRONDeviceDateSettingsKey];


NSMutableArray *deviceSettings = [[NSMutableArray alloc] init];
[deviceSettings addObject:personalSettings]; // Same as previous section
[deviceSettings addObject:timeFormatSettings];
[deviceSettings addObject: dateFormatSettings];

// Updating device settings
peripheralConfig.deviceSettings = deviceSettings;
```

c) Distance Settings and Sleep Settings:

```
// Distance Format
NSDictionary *distanceFormatSettings = @{OMRONDeviceDistanceSettingsUnitKey :
@(OMRONDeviceDistanceUnitKilometer) };

NSMutableDictionary *distanceSettings = [[NSMutableDictionary alloc] init];
[distanceSettings setObject:distanceFormatSettings
forKey:OMRONDeviceDistanceSettingsKey];

// Sleep Settings
NSDictionary *sleepTimeSettings =
@{ OMRONDeviceSleepSettingsAutomaticKey: @(OMRONDeviceSleepAutomaticOn),
OMRONDeviceSleepSettingsAutomaticStartTimeKey : @"20",
OMRONDeviceSleepSettingsAutomaticStopTimeKey : @"8"};

NSMutableDictionary *sleepSettings = [[NSMutableDictionary alloc] init];
sleepSettings setObject:sleepTimeSettings forKey:OMRONDeviceSleepSettingsKey];


[deviceSettigs addObject:distanceSettings];
[deviceSettigs addObject:sleepSettings];
```

Configuration for Body Composition Devices:

a) Device and personal settings

```
// Weight Settings (Personal settings)

NSDictionary *weightPersonalSettings = @{ OMRONDevicePersonalSettingsWeightDCIKey:
@(OMRONDevicePersonalSettingsWeightDCINotAvailable)};

NSDictionary *settings = @{
OMRONDevicePersonalSettingsUserHeightKey: <HEIGHT (cm)>,
OMRONDevicePersonalSettingsUserDateOfBirthKey: <DateOfBirth (yyyymmdd)>,
OMRONDevicePersonalSettingsUserGenderKey: <Gender
(OMRONDevicePersonalSettingsUserGenderTypeMale)>,
OMRONDevicePersonalSettingsWeightKey: <WeightKey (weightPersonalSettings)> };

NSMutableDictionary *personalSettings = [[NSMutableDictionary alloc] init];
[personalSettings setObject:settings forKey:OMRONDevicePersonalSettingsKey];

// Weight Settings
// Add other weight common settings if any
NSDictionary *weightCommonSettings = @{ OMRONDeviceWeightSettingsUnitKey :
@(OMRONDeviceWeightUnitLbs) };
NSMutableDictionary *weightSettings = [[NSMutableDictionary alloc] init];
[weightSettings setObject:weightCommonSettings forKey:OMRONDeviceWeightSettingsKey];
NSMutableArray *deviceSettigs = [[NSMutableArray alloc] init];
[deviceSettigs addObject:personalSettings];
[deviceSettigs addObject:weightSettings];

// Set Device Settings
peripheralConfig.deviceSettings = deviceSettigs;


 // Set Configuration to New Configuration
[[OmronPeripheralManager sharedManager] setConfiguration:peripheralConfig];
```

b) Weight unit Setting

```
// Weight Settings
// Add other weight common settings if any
NSDictionary *weightCommonSettings = @{OMRONDeviceWeightSettingsUnitKey :
@(OMRONDeviceWeightUnitLbs) };
NSMutableDictionary *weightSettings = [[NSMutableDictionary alloc] init];
[weightSettings setObject:weightCommonSettings
forKey:OMRONDeviceWeightSettingsKey];

NSMutableArray *deviceSettings = [[NSMutableArray alloc] init]; [deviceSettings
addObject:personalSettings]; // Same as previous section [deviceSettigs
addObject:weightSettings];

// Updating device settings
peripheralConfig.deviceSettings = deviceSettings;
```

c) Guest mode weight Only configuration for Body composition monitor

```
// Weight Settings
// Add other weight common settings if any
NSDictionary *weightCommonSettings = @{ OMRONDeviceWeightSettingsUnitKey :
@(OMRONDeviceWeightUnitLbs), OMRONDeviceWeightSettingsWeightOnlyKey :
@(OMRONDeviceWeightOnlyOn), OMRONDeviceWeightSettingsOneTimeMeasurementKey:
@(OMRONDeviceWeightOneTimeMeasurementOn) };
NSMutableDictionary *weightSettings = [[NSMutableDictionary alloc] init];
[weightSettings setObject:weightCommonSettings
forKey:OMRONDeviceWeightSettingsKey];

NSMutableArray *deviceSettings = [[NSMutableArray alloc] init]; [deviceSettings
addObject:personalSettings]; // Same as previous section [deviceSettigs
addObject:weightSettings];

// Updating device settings
peripheralConfig.deviceSettings = deviceSettings;
```

d) Guest mode weight Only configuration for Body composition monitor

```
// Weight Settings
// Add other weight common settings if any
NSDictionary *weightCommonSettings = @{ OMRONDeviceWeightSettingsUnitKey :
@(OMRONDeviceWeightUnitLbs), OMRONDeviceWeightSettingsWeightOnlyKey :
@(OMRONDeviceWeightOnlyOff), OMRONDeviceWeightSettingsOneTimeMeasurementKey:
@(OMRONDeviceWeightOneTimeMeasurementOn) };
NSMutableDictionary *weightSettings = [[NSMutableDictionary alloc] init];
[weightSettings setObject:weightCommonSettings
forKey:OMRONDeviceWeightSettingsKey];

NSMutableArray *deviceSettings = [[NSMutableArray alloc] init]; [deviceSettings
addObject:personalSettings]; // Same as previous section [deviceSettigs
addObject:weightSettings];

// Updating device settings
peripheralConfig.deviceSettings = deviceSettings;
```

5. Discovering Omron Connected Devices

Partner application can start connecting to Omron Connected Devices once configurations are set in OmronPeripheralManager. To begin discovering devices, application need to start the OmronPeripheralManager using below function call.

```
[[OmronPeripheralManager sharedManager] startManager];
```

Partner application can begin discovering for Omron Connected Devices supported by the OmronConnectivityLibrary.framework.

OmronPeripheralManager class needs to be used to discover Omron Connected Devices. A list of OmronPeripheral will be returned upon success. In case of any failures an error object will be returned.

```
[[OmronPeripheralManager sharedManager]
startScanPeripheralsWithCompletionBlock:^(NSArray *retrievedPeripherals,
NSError *error) {

}];
```

NOTE 1: If partner application is not using any device filters all Omron Connected Devices will be discovered by OmronPeripheralManager. If device filters are provided OmronPeripheralManager will return only filtered Omron Device model. In either of these cases, multiple devices of same type will be returned if available.

NOTE 2: This function has a default timeout of 60 seconds. OmronPeripheralManager will keep scanning for Omron Connected devices for 60 seconds and return list of devices discovered during this process in real-time. If no devices are available it will give a timeout error object.

NOTE 3: When required, partner application can stop scanning for Omron Connected devices using the below function.

```
// Stop Scanning with completion block
[[OmronPeripheralManager sharedManager]
stopScanPeripheralsWithCompletionBlock:^(NSError *error) {

}];
```

Or

```
// Stop Scanning
[[OmronPeripheralManager sharedManager] stopScanPeripherals];
```

6. Connecting Omron Connected Devices

Partner application can start connecting to discovered Omron Connected Devices by OmronPeripheralManager. Following library function call facilitates connecting to a particular Omron Connected device.

```
[[OmronPeripheralManager sharedManager] connectPeripheral:peripheral
withCompletionBlock:^(OmronPeripheral *peripheral, NSError *error) {


}];
```

For devices that supports more than 1 users, following methods needed to be used since a user number requires to be passed to connect the device with the App for a particular user.

List of available users can be retrieved from device settings that will also provide the information about registered users as well. See Appendix that has the sample output for the same.

`OMRONDeviceSettingsKey` contains the information about available users and registered users under `OMRONDeviceSettingsAvailableUsersKey` and `OMRONDeviceSettingsRegisteredUsersKey` respectively.

This information can be used while connecting the App with the device for a particular user.

```
[[OmronPeripheralManager sharedManager] connectPeripheral:peripheral
withWait:YES withCompletionBlock:^(OmronPeripheral *peripheral, NSError
*error) {

}];
```

On the completion of the above method, the following method requires to be used with the user number that will connect the App with the device for that particular user.

```
[[OmronPeripheralManager sharedManager]
resumeConnectPeripheralWithUser:selectUser
withCompletionBlock:^(OmronPeripheral *peripheral, NSError *error) {

}];
```

or

```
[[OmronPeripheralManager sharedManager]
resumeConnectPeripheralWithUsers:selectUsers
withCompletionBlock:^(OmronPeripheral *peripheral, NSError *error) {

}];
```

This will initiate a Bluetooth pairing request with the Omron Connected Device and smartphone. Once Bluetooth Pairing request is accepted by the end user, the device is connected to smartphone and OmronPeripheralManager is ready to communicate with device. OmronPeripheralManager returns the Omron Connected Device details to Partner application in form of OmronPeripheral object. If any failures happen during connectivity an error object is returned to partner application.

OmronPeripheralManager will setup the Omron Device in this step by updating the date and time in device.

The following method can be used to end the connection if needed. After initiating a connection using `connectPeriperhal:withWait:`withCompletionBlock: method, following method can be used to end the connection. An example

scenario to use this will be - if connection was started for user ‘2’ and that user is already registered with the device.

```
[[OmronPeripheralManager sharedManager]
endConnectPeripheralWithCompletionBlock:^(OmronPeripheral *peripheral,
NSError *error) {

}];
```

Partner applications can act in two different scenarios while scanning for Omron Connected devices.

a. Explicit Connection: Partner application can keep scanning for devices and keep track of these and later connect to one of these from list.
b. Implicit Connection: Partner application can choose to connect to first discovered device.

7. Transferring Vital Data from Omron Connected Devices

Partner applications can transfer data from already paired Omron Connected Devices. The application has to pass the user number for which data need to be transferred from blood pressure monitor or body composition device. Each Omron Blood Pressure monitor and body composition device has defined number of user types. Some blood pressure monitor devices are single user device and others are two user devices. Body composition devices are four user devices. The device configuration retrieved in step V (2) provides information about this. Partner application need to pass in the required user type for data transfer.

The different data available from Omron Connected Devices are

- Blood Pressure
- Activity
- Sleep
- Records
- Weight and Body Composition data
- Wheeze Data
- Pulse Oximeter Data
- Temperature

Once OmronConnectivityLibrary transfers all data from Omron Connected Devices, the unsent data flag on the device is cleared. This means,

OmronConnectivityLibrary will not be able to read the already transferred data again.

Partner application has to pause the data transfer till all data is saved securely and then end connection with Omron Connected Devices. For this purpose, below two functions need to be used one after the other. The first function starts data transfer and shares transferred data with Partner app. Now the partner application can save this data and invoke the second function. The second function confirms that the data transfer is completed successfully and clears the unsent data flag on device. **If this approach is not implemented properly, then it can lead to data loss.**

Start Data Transfer and Pause

```
OmronPeripheral *peripheral = [[OmronPeripheral alloc] initWithLocalName:localName
andUUID:UUID];

[[OmronPeripheralManager sharedManager] startDataTransferFromPeripheral:peripheralLocal
withUser:_selectUser withWait:YES withCompletionBlock:^(OmronPeripheral *peripheral, NSError
*error) {

}];
```

or

```
OmronPeripheral *peripheral = [[OmronPeripheral alloc] initWithLocalName:localName
andUUID:UUID];

[[OmronPeripheralManager sharedManager] startDataTransferFromPeripheral:peripheralLocal
withUsers:selectUsers withWait:YES withCompletionBlock:^(OmronPeripheral *peripheral, NSError
*error) {

}];
```

End Data Transfer and clear unsent flags for data:

```
[[OmronPeripheralManager sharedManager]
endDataTransferFromPeripheralWithCompletionBlock:^(OmronPeripheral *peripheral, NSError *error)
{

}];
```

8. Recording Temperature Data from Omron Connected Devices using Audio

For OMRON devices that support Audio communication, applications can transfer reading from device using recording function. There will not be Bluetooth communication for such devices. Microphone access need to be granted by application to record data and required iOS frameworks need to be added to project to support audio communication.

Below are the details for connecting using MC-280B Audio device. OmronPeripheral is defined using constant identifiers and recording is initiated. onSignalStrength gives signal strength of audio signal from device. Once recording is transferred to app, data can be fetched using library function described in next section. Recording need to stopped once data is received explicitly. Device will keep sending data till recording is stopped by the application.

Refer appendix for error codes related to audio connectivity.

```
// Configure OMRON Peripheral with Local Name per specification
OmronPeripheral *peripheral = [[OmronPeripheral alloc]
initWithLocalName:OMRONThermometerMC280B andUUID:@""];

// Start Recording
[[OmronPeripheralManager sharedManager] startRecording:peripheral
                                        onSignalStrength:^(double signal) {

} withCompletionBlock:^(OmronPeripheral *peripheral, NSError *error) {

}];
```

```
// Stop recording
[[OmronPeripheralManager sharedManager]
stopRecordingWithCompletionBlock:^(OmronPeripheral *peripheral, NSError *error) {


}];
```

9. Retrieve Vital Data

Partner application uses OmronPeripheral class to retrieve Vital Data for a selected user. Refer instruction manual of Omron device to understand different data types available from devices. The different data available from Omron Connected Devices are

- Blood Pressure
- Activity
- Sleep
- Records
- Weight
- Wheeze
- Pulse Oximeter
- Temperature

Below section shows how Partner applications can retrieve vital data once paired and data is transferred. The below will be used on the completion of startDataTransferFromPeripheral:withUser:withWait:withCompletionBlock

```
[peripheral getVitalDataWithUser:_selectUser withCompletionBlock:^(NSMutableDictionary *vitalData,
NSError *error) {

for (NSString *key in vitalData.allKeys) {

        if([key isEqualToString:OMRONVitalDataBloodPressureKey]) {
            // Blood Pressure Data
        }else if([key isEqualToString:OMRONVitalDataActivityKey]) {
            // Activity Data
        }else if([key isEqualToString:OMRONVitalDataSleepKey]) {
            // Sleep Data
        }else if([key isEqualToString:OMRONVitalDataRecordsKey]) {
            // Records Data
        }else if([key isEqualToString:OMRONVitalDataWeightKey]) {
            // Weight Data
        }else if([key isEqualToString:OMRONVitalDataWheezeKey]) {
            // Wheeze Data
        }else if([key isEqualToString:OMRONVitalDataPulseOximeterKey]) {
            // Pulse Oximeter
        }else if([key isEqualToString:OMRONVitalDataTemperatureKey]) {
            // Temperature
        }

    }
}];
```

VITAL DATA:
Omron Connected Activity Devices support Activity data, Sleep data and Records Data in addition to Blood Pressure Data.

KEY	DESCRIPTION	DEVICE TYPE
OMRONVitalDataBloodPressureKey	Blood Pressure	Blood Pressure
OMRONVitalDataActivityKey	Activity Data	Activity
OMRONVitalDataSleepKey	Sleep Data	Activity
OMRONVitalDataRecordKey	Records	Activity
OMRONVitalDataWeightKey	Weight	Body Composition
OMRONVitalDataWheezeKey	Wheeze	Wheeze
OMRONVitalDataPulseOximeterKey	Oxygen Level	Pulse Oximeter
OMRONVitalDataTemperatureKey	Temperature data	Thermometer

BLOOD PRESSURE:

Vital Data contains blood pressure readings transferred from Omron Connected Device. Use key OMRONVitalDataBloodPressureKey to get Blood Pressure data from Vital Data list.

KEY	DESCRIPTION	UNIT
OMRONVitalDataSystolicKey	Systolic Blood Pressure	mmHg
OMRONVitalDataDiastolicKey	Diastolic Blood Pressure	mmHg
OMRONVitalDataPulseKey	Pulse	bpm
OMRONVitalDataMovementFlagKey	Movement Error Symbol	-
OMRONVitalDataCuffFlagKey (for display)	Cuff Wrap Guide	0 – unfit 1 - fit
OMRONVitalDataConsecutiveMeasurementKey	TruRead Index	-
OMRONVitalDataArtifactDetectionKey	Number of detected artifact	-
OMRONVitalDataIrregularFlagKey	Irregular Heart Beat Symbol	-
OMRONVitalDataMeasurementStartDateKey	Date Time Unix Timestamp (UTC)	-

OMRONVitalDataCuffWrapDetectionFlagKey (not for display)	Cuff Wrap Guide	0 – unfit 1 - fit
OMRONVitalDataIrregularHeartBeatCountKey	No of irregular heartbeat	-
OMRONVitalDataMeasurementModeKey	Blood pressure measurement mode	-
OMRONVitalDataDisplayedErrorCodeNightModeKey	Blood pressure measurement error code when using NightView device	-

The data object available with OMRONVitalDataBloodPressureKey will be a list of blood pressure readings having measurements like systolic, diastolic, pulse, date time and so on. Each of these data can be retrieved using the keys mentioned in OmronDefines.h under section OMRON Connectivity Library Blood Pressure Data Keys. Refer connectivity constant to identify all blood pressure data properties.

TruRead Mode for only BP786/CAN, BP786N/CANN:
Only BP786/CAN, BP786N/CANN support TruRead feature. The TruRead Mode takes 3 consecutive measurements. The monitor will inflate, take a measurement, and deflate - 3 times, separated by a short interval between each measurement. The TruRead Mode is set "OFF" by default. The available options for short interval are - set 15, 30, 60, or 120 seconds. Omron Blood Pressure monitor shows these 3 readings as a single reading in device.

When Partner application uses OmronConnectivityLibrary.framework to retrieve vital data, it should consider this mode of Omron Blood Pressure Monitor. A flag value is associated with each vital data returned from the device. Partner application can use key OMRONVitalDataConsecutiveMeasurementKey to identify this. Below tables explains the usage of this flag in Vital data.

TruRead Mode OFF

VALUE	DESCRIPTION
Not Available	Not a TruRead Vital Data

TruRead Mode ON

VALUE	DESCRIPTION
1	First Vital Data
2	Second Vital Data
3	Third Vital Data

Averaging Logic for TruRead Vital Data:

Three readings a, b and c needs to be averaged.
Average = (a+b+c)/3, where average is rounded to the nearest integer

Rounding Logic: If the point after decimal is greater or equal to 5, it is rounded to the next higher integer. If the point after decimal is less than 5, it is rounded to the previous integer.
For Example,
96.5 displayed as 97
96.4 displayed as 96

Blood Pressure Measurement mode (OMRONVitalDataMeasurementModeKey):

VALUE	DESCRIPTION
0	Normal Mode
1	AFiB Mode
2	Night + Time designation (2 am, 4 am, so on)
3	Night + Elapsed time (4 hours after setting Nocturnal mode)
4	Night mode measurement is started by two settings at the same time
5	TruRead Mode

ACTIVITY:

Vital Data contains activity data transferred from Omron Connected Device. Use key OMRONVitalDataActivityKey to get activity data from Vital Data list.

The data object available with OMRONVitalDataActivityKey will be list of activity data for days. Each object in the list is a dictionary of different activity types like Steps, Calories, Aerobic steps and Distance available in activity devices. Each of these activity types will have list of activity data like measurement, start/end date, and sequence number.
Keys are mentioned in OmronDefines.h under section OMRON Connectivity Library Activity Data Keys

Different type of Activity data available are listed below

ACTIVITY TYPE	DESCRIPTION	UNIT
OMRONActivityStepsPerDay	Steps for a day	count
OMRONActivityAerobicStepsPerDay	Aerobic steps for a day	count
OMRONActivityWalkingCaloriesPerDay	Calories burnt in a day	kcal
OMRONActivityDistancePerDay	Distance covered in a day	km
OMRONActivityNormalStepsPerDay	Normal steps in a day	count
OMRONActivityJoggingStepsPerDay	Jogging steps in a day	count
OMRONActivityFastStepsPerDay	Fast steps in a day	count

Each of activity types contains below data

KEY	DESCRIPTION
OMRONActivityDataStartDateKey	Start Date Time Unix Timestamp (UTC)
OMRONActivityDataEndDateKey	End Date Time Unix Timestamp (UTC)
OMRONActivityDataMeasurementKey	Measurement
OMRONActivityDataSequenceKey	Sequence
OMRONActivityDataDividedDataKey	Hourly data (list)

Each of divided data contains below data

KEY	DESCRIPTION
OMRONActivityDividedDataStartDateKey	Start Date Time Unix Timestamp for Hourly data (UTC)
OMRONActivityDividedDataMeasurementKey	Measurements in Hourly data
OMRONActivityDividedDataPeriodTimeKey	Period in Hourly data
OMRONActivityDividedDataMeasurementDetailsKey	Measurement details

Note: The data transferred is duplicated if transfer happens more than once in same day. It has to be made sure that duplicates are processed appropriately. OMRONActivityDataSequenceKey is used to check for duplicity of activity data for a day. The value corresponding to this key is unique.

SLEEP:

Vital Data contains sleep data transferred from Omron Connected Device. Use key OMRONVitalDataSleepKey to get sleep data from Vital Data list.

The data object available with OMRONVitalDataSleepKey will be a dictionary of different sleep types like Sleep Time, Wakeup Time, Total Sleep Time, Body Movement Level and so on.

Keys are mentioned in OmronDefines.h under section OMRON Connectivity Library Sleep Data Keys

Available data present in sleep are listed below

KEY	DESCRIPTION
OMRONSleepDataStartDateKey	Start Date Time Unix Timestamp (UTC)
OMRONSleepDataEndDateKey	End Date Time Unix Timestamp (UTC)
OMRONSleepTimeInBedKey	Time in bed – Unix Timestamp (UTC)
OMRONSleepSleepOnsetTimeKey	Sleep Time – Unix Timestamp (UTC)
OMRONSleepWakeTimeKey	Wakeup Time – Unix Timestamp (UTC)
OMRONSleepTotalSleepTimeKey	Total Sleep Time in minutes
OMRONSleepSleepEfficiencyKey	Efficiency of sleep in percentage
OMRONSleepArousalDuringSleepTimeKey	Minutes awake
OMRONSleepBodyMotionLevelKey	List of body movement level during sleep

Body movement level:

Value	DESCRIPTION
0	Level 0
1	Level 1
2	Level 2
3	Not measured

RECORDS:

The Record Data is a list containing different date time. Each date time entry denotes the time a record was saved in Omron device. Keys are mentioned in OmronDefines.h under section OMRON Connectivity Library Records Data Keys

WEIGHT DATA:

Vital Data contains weight data transferred from Omron Connected Device. Use key OMRONVitalDataWeightKey to get Weight data from Vital Data list.

The data object available with OMRONVitalDataWeightKey will be a list of weight readings having measurements like weight, body fat, BMI, skeletal muscle and so on. Each of these data can be retrieved using the keys mentioned in OmronDefines.h under section OMRON Connectivity Library Weight Data Keys

KEY	DESCRIPTION	Range/Unit
OMRONWeightDataStartDateKey	Measurement Start Date	timeStamp
OMRONWeightDataSequenceKey	Sequence Number	Int
OMRONWeightDataUserIdKey	User id	1-4
OMRONWeightKey	Weight	2 – 150 (Kg)
OMRONWeightBodyFatLevelClassificationKey	Body Fat Classification	-
OMRONWeightBodyFatPercentageKey	Body Fat Percentage	5.0 - 60.0
OMRONWeightRestingMetabolismKey	Resting Metabolism	385 - 3999
OMRONWeightSkeletalMusclePercentageKey	Skeletal Muscle Percentage	5.0 – 50.0
OMRONWeightBMIKey	Body Mass Index (BMI)	7.0 – 90.0
OMRONWeightVisceralFatLevelKey	Visceral Fat	Upto 30
OMRONWeightVisceralFatLevelClassificationKey	Visceral Fat Classification	Upto 4 Levels
OMRONWeightSkeletalMuscleLevelClassificationKey	Skeletal Muscle Classification	-

WHEEZE DATA:

Vital Data contains wheeze data transferred from Omron Connected Device. Use key OMRONVitalDataWheezeKey to get Wheeze data from Vital Data list.

The data object available with OMRONVitalDataWheezeKey will be a list of wheeze readings having measurements like wheeze detected, error, noise and so on. Each of these data can be retrieved using the keys mentioned in OmronDefines.h under section OMRON Connectivity Library Wheeze Data Keys

KEY	DESCRIPTION	Range/Unit
OMRONWheezeDataStartDateKey	Measurement Start Date	timeStamp
OMRONWheezeDataSequenceKey	Sequence Number	Int
OMRONWheezeDataUserIdKey	User id	1
OMRONWheezeKey	Wheeze data	0 : Detected 1 : Not detected 2 : Measurement Error
OMRONWheezeErrorNoiseKey	Excessive Noise Error	0 : No error 1 : Error
OMRONWheezeErrorDecreaseBreathingSoundLevelKey	Decrease in breathing sound error	0 : No error 1 : Error
OMRONWheezeErrorSurroundingNoiseKey	Environmental noise error	0 : No error 1 : Error

PULSE OXIMETER DATA:

Vital Data contains pulse oximeter data transferred from Omron Connected Device. Use key OMRONVitalDataPulseOximeterKey to get pulse oximeter data from Vital Data list.

The data object available with OMRONVitalDataPulseOximeterKey will be a list of pulse oximeter readings having measurements like oxygen level, pulse rate, pulse amplitude index. Each of these data can be retrieved using the keys mentioned in OmronDefines.h under section OMRON Connectivity Library Pulse Oximeter Data Keys

KEY	DESCRIPTION	Range/Unit
OMRONPulseOximeterDataStartDateKey	Measurement Start Date	timeStamp
OMRONPulseOximeterDataSequenceKey	Sequence Number	Int
OMRONPulseOximeterDataUserIdKey	User id	1
OMRONPulseOximeterSPO2LevelKey	Blood oxygen level	Percentage
OMRONPulseOximeterPulseRateKey	Pulse rate	bpm
OMRONPulseOximeterAmplitudeKey	Pulse amplitude index	-

TEMPERATURE DATA:

Vital Data contains temperature data transferred from Omron Connected Device. Use key OMRONVitalDataPulseTemperatureKey to get temperature data from Vital Data list.

The data object available with OMRONVitalDataTemperatureKey will be a list of temperature readings having measurements like temperature as shown in LCD, temperature in Celsius, temperature unit, temperature. Each of these data can be retrieved using the keys mentioned in OmronDefines.h under section OMRON Connectivity Library Temperature Data Keys

KEY	DESCRIPTION	Range/Unit
OMRONTemperatureDataStartDateKey	Measurement Start Date	timeStamp
OMRONTemperatureDataSequenceKey	Sequence Number	Int
OMRONTemperatureDataUserIdKey	User id	1
OMRONTemperatureKey	Temperature value (LCD display)	-
OMRONTemperatureCelsiusKey	Temperature value in Celsius	bpm
OMRONTemperatureUnitKey	Unit of Temperature	-Celsius / Fahrenheit
OMRONTemperatureLevelKey	Temperature level	1 : High 0 : Low

10. Disconnecting Omron Connected Devices (optional)

Partner application can choose to explicitly disconnect from Omron Connected Devices when required using the below Library API call with the help of OmronPeripheral object returned by library in above steps.

```
[[OmronPeripheralManager sharedManager] disconnectPeripheral:omronPeripheral
withCompletionBlock:^(OmronPeripheral *peripheral, NSError *error) {

}];
```

NOTE: This library API is required by Partner application only if they need to explicitly disconnect from an Omron Connected Device during an ongoing connection communication.

11. Updating Device Settings

Partner application can update the device time format setting (24Hr/12Hr), add a maximum of 5 alarms and other settings as described in Section V(3) d. The time settings and alarms should be part of deviceSettings configuration in the OmronPeripheralManagerConfig. This is described in Step V (3).

```
OmronPeripheralManagerConfig *peripheralConfig = [[OmronPeripheralManager sharedManager]
getConfiguration];

// Add all required settings for the peripheral
peripheralConfig.deviceSettings = newConfig;

// Set Configuration to New Configuration
[[OmronPeripheralManager sharedManager] setConfiguration:peripheralConfig];

// Start Updating Device
[[OmronPeripheralManager sharedManager] updatePeripheral:peripheral
withCompletionBlock:^(OmronPeripheral *peripheral, NSError *error) {

 }];
```

In case of devices that support more than 1 user, following method should be used that has a new parameter to send the selected user number, this is to update the settings for the particular user only.

```
OmronPeripheralManagerConfig *peripheralConfig = [[OmronPeripheralManager
sharedManager] getConfiguration];

// Add all required settings for the peripheral
peripheralConfig.deviceSettings = newConfig;

// Set Configuration to New Configuration
[[OmronPeripheralManager sharedManager] setConfiguration:peripheralConfig];

// Start Updating Device
[[OmronPeripheralManager sharedManager] updatePeripheral:peripheral
withUser:selectedUser withCompletionBlock:^(OmronPeripheral *peripheral,
NSError *error) {

}];

// or start Updating Device with users
[[OmronPeripheralManager sharedManager] updatePeripheral:peripheral
withUsers:selectedUsers withCompletionBlock:^(OmronPeripheral *peripheral,
NSError *error) {

}];
```

Please refer table under Appendix for the setting that you can update for each device type.

12. Additional Features (optional)

a) Device Information

Library provides additional device information like device local name and device UUID. Partner application can request for this detail after every Pair (Connect) or Transfer process. Different supported keys, which provide data, are defined in OmronDefines.h.

```
[peripheral getDeviceInformationWithCompletionBlock:^(NSMutableDictionary
*deviceInfo, NSError *error) {

}];
```

Alternatively

```
[peripheral getDeviceInformation];
```

KEY	DESCRIPTION	DEVICE TYPE
OMRONDeviceInformationDisplayNameKey	Display name	All
OMRONDeviceInformationIdentityNameKey	Identity name	All
OMRONDeviceInformationLocalNameKey	Local name	All
OMRONDeviceInformationSerialIdKey	Serial Id	All
OMRONDeviceInformationUUIDKey	UUID	All
OMRONDeviceInformationiBeaconProximityUUIDKey	Proximity UUID of region	Support iBeacon
OMRONDeviceInformationiBeaconMajorValueKey	Major value	Support iBeacon
OMRONDeviceInformationiBeaconMinorValueKey	Minor value	Support iBeacon

Refer Apple documentation on iBeacon for more details
https://developer.apple.com/ibeacon/

b) Device Settings

Device settings can be fetched using the following method that contains device and personal settings.

```
[peripheral getDeviceSettings];
```

The above method will provide the information about:

- Weight Unit Settings (For Body Composition)
- Time Settings (For Activity and Blood Pressure Devices)
- Date Settings (For Activity and Blood Pressure Devices)
- Distance Settings (For Activity Devices)
- Sleep Settings (For Activity Devices)
- iBeacon Settings
- Personal Settings

Data received from above method will include all information related to device settings, available users, registered users and user's personal settings. See Appendix VIII 3 (c) I. User's personal settings are under key `OmronDevicePersonalSettingsKey`,

Display Enable/Disable and Priority Settings (Body Composition Device):

These personal settings also have the information about display settings i.e. enable/disable and priority of display settings. Enable/Disable keys provides the information about different items of device if the display of those are enable or disable and the priority order tells the order in which these will be displayed on the device. See Appendix VIII 3 (c) II. Following tables contains the required information about these keys.

KEY	DESCRIPTION
OMRONDevicePersonalSettingsWeightDisplayPriorityBodyFatKey	Display priority – Body Fat
OMRONDevicePersonalSettingsWeightDisplayPriorityVisceralFatLevelKey	Display priority – Visceral Fat
OMRONDevicePersonalSettingsWeightDisplayPrioritySkeletalMuscleLevelKey	Display priority – Skeletal Fat
OMRONDevicePersonalSettingsWeightDisplayPriorityRestingMetabolismKey	Display priority – Resting Metabolism
OMRONDevicePersonalSettingsWeightDisplayPriorityBMIKey	Display priority – BMI
OMRONDevicePersonalSettingsWeightDisplayEnableBodyFatKey	Display enable – Body Fat
OMRONDevicePersonalSettingsWeightDisplayEnableVisceralFatLevelKey	Display enable – Visceral Fat
OMRONDevicePersonalSettingsWeightDisplayEnableSkeletalMuscleLevelKey	Display enable – Skeletal Fat
OMRONDevicePersonalSettingsWeightDisplayEnableRestingMetabolismKey	Display enable – Resting Metabolism
OMRONDevicePersonalSettingsWeightDisplayEnableBMIKey	Display enable – BMI

To get settings data for a particular user, following method can be used that requires a user number as well. See Appendix VIII 3(c) II for sample data.

```
[peripheral getDeviceSettingsWithUser:selectedUser];
```

Alternatively, following method can be used that has a completion block to get device settings are received or an error in case something went wrong.

```
[peripheral getDeviceSettingsWithUser:selectedUser withCompletionBlock:^(id
deviceSettings, NSError *error) {

}];
```

c) Bluetooth State Change Information

Application can listen to Bluetooth events when communicating with Omron Connected Devices. Two additional features provided by Library are

I. Bluetooth State Change Notification
Application can listen to Bluetooth state changes. Partner application need to register to notification like below.

```
// Notification Listener for BLE State Change
[[NSNotificationCenter defaultCenter] addObserver:self

selector:@selector(centralManagerDidUpdateStateNotification:)

name:OMRONBLECentralManagerDidUpdateStateNotification
                                                object:nil];

#pragma mark - CoreBluetooth Notifications

- (void)centralManagerDidUpdateStateNotification:(NSNotification
*)aNotification {

OMRONBLEBluetoothState bluetoothState =
(OMRONBLEBluetoothState)[aNotification.object unsignedIntegerValue] ;

    if(bluetoothState == OMRONBLEBluetoothStateOn) {
        NSLog(@"%@",  @"Bluetooth is currently powered on.");
    }else if(bluetoothState == OMRONBLEBluetoothStateOff) {
        NSLog(@"%@",  @"Bluetooth is currently powered off.");
    }else if(bluetoothState == OMRONBLEBluetoothStateUnknown) {
        NSLog(@"%@",  @"Bluetooth is in unknown state");
    }else if(bluetoothState == OMRONBLEBluetoothStateUnauthorized) {
        NSLog(@"%@",  @"Bluetooth is in unauthorized state");
    }

}
```

II. Bluetooth Device state changes

Application gets notified when the Bluetooth connected device's state changes.

```
[[OmronPeripheralManager sharedManager] onConnectStateChangeWithCompletionBlock:^(int state)
{

    if (state == OMRONBLEConnectionStateConnecting) {
        NSLog(@"Connecting");
    } else if (state == OMRONBLEConnectionStateConnected) {
        NSLog(@"Connected");
    } else if (state == OMRONBLEConnectionStateDisconnecting) {
        NSLog(@"Disconnecting");
    } else if (state == OMRONBLEConnectionStateDisconnect) {
        NSLog(@"Disconnected");
    }
```

d) Get Library Version

Application can check the current version of the library using below.

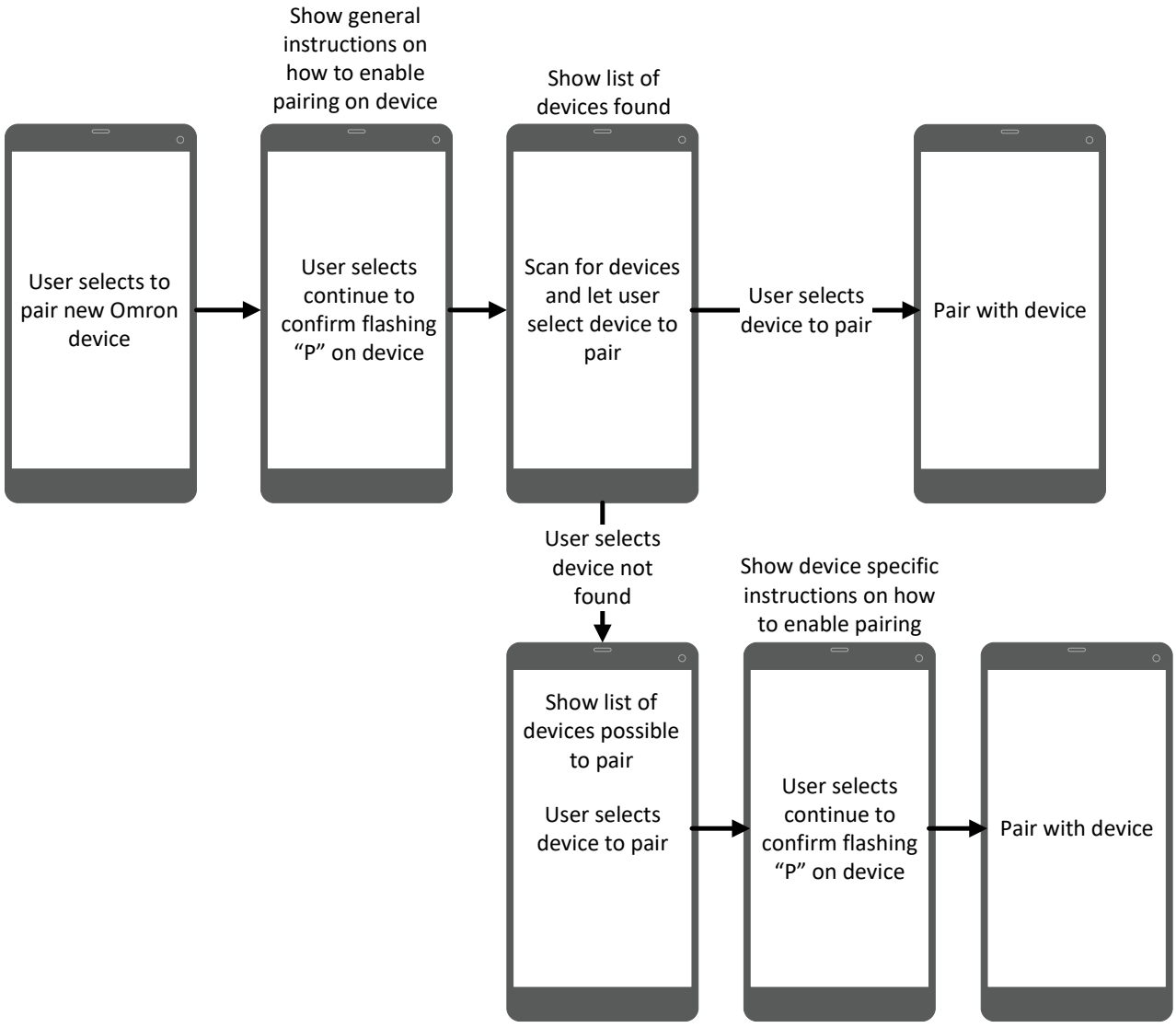
```
[[OmronPeripheralManager sharedManager] libVersion];
```

VII. Implementation Strategies

It is mandatory for Partner applications to implement the OmronConnectivityLibrary framework following the implementation strategies. For pairing partner may select one of two ways to interact with Omron Connected Devices. Resources for specific device pairing and data transfer instructions can be found in the resources directory.

1. Pairing: Scan all Omron Connected Devices

Partner application can scan for all Omron Connected Devices. Once devices are discovered user can then choose to connect to desired device. If device is not found user can select device not found to select a specific device. OmronPeripheralManager can now interact with device and perform Vital data transfer.



Identifying Device Model/No of Users available for the Device Model:

Step 2 in "OmronConnectivityLibrary Framework Integrations" describes how a Partner application can retrieve the list of supported Omron Blood Pressure

Monitors.

Once partner application successfully pairs/connects a device to application, it can check with OmronConnectivityLibrary framework to identify the device model and number of users available in the device. An OmronPeripheral object is returned to Partner application after successful pairing and this object has below keys, which will help identify details of device.

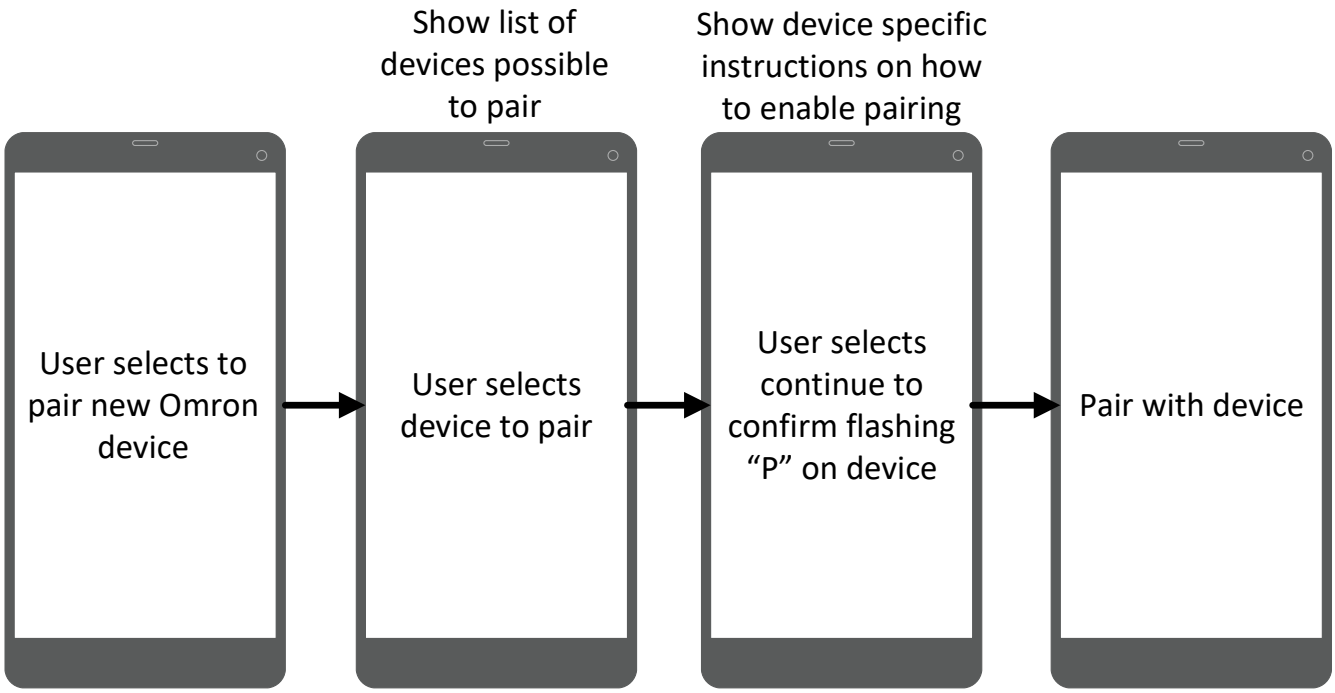
KEY
OMRONBLEConfigDeviceGroupIDKey
OMRONBLEConfigDeviceGroupIncludedGroupIDKey

Below function can then be used to retrieve device configuration.

```
OmronPeripheralManagerConfig *peripheralConfig = [[OmronPeripheralManager sharedManager]
getConfiguration];
        NSMutableDictionary *deviceConfig = [peripheralConfig
retrievePeripheralConfigurationWithGroupId: [peripheral
valueForKey:OMRONBLEConfigDeviceGroupIDKey] andGroupIncludedId: [peripheral
valueForKey:OMRONBLEConfigDeviceGroupIncludedGroupIDKey]];
```

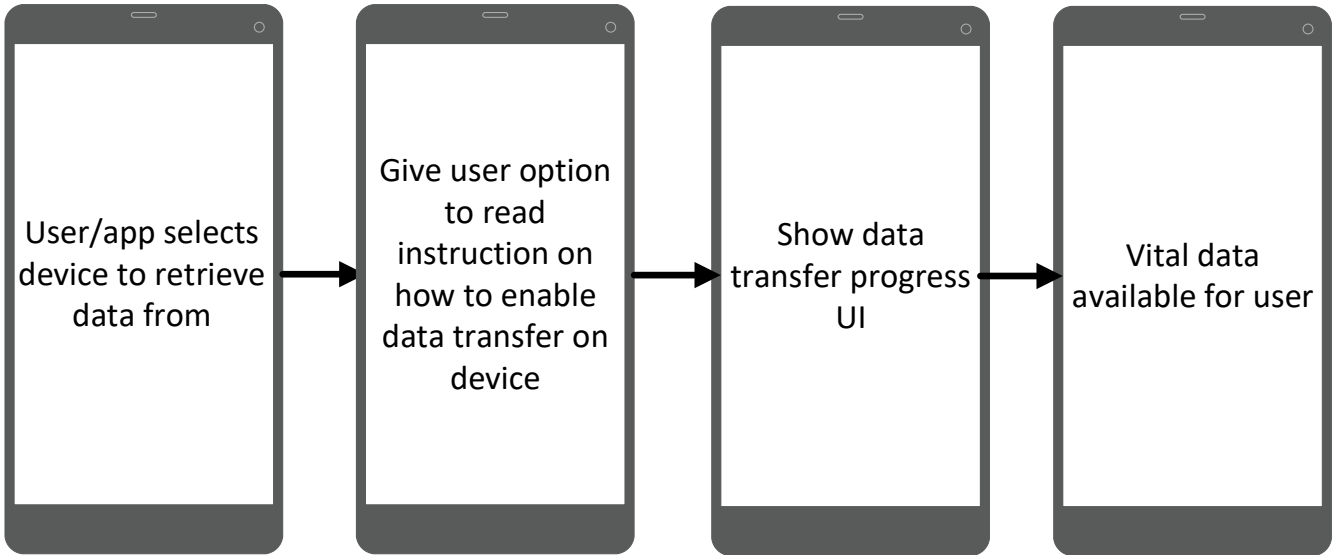
2. Pairing: Scan for only selected Omron Connected Devices

Partner application can choose to scan only for selected Omron Connected Device model. This will allow OmronPeripheralManager to filter discovering Omron Connected Devices.



3. Transferring data from already paired Devices

Partner application should store paired devices information (one or many). This will help to Transfer/ Auto Transfer data from the Paired devices. User can select which from device to transfer data in case of more than one paired device.



VIII. Error Handling and Connectivity Scenarios

1. Error Handling

When partner application faces a connectivity issue with Omron Connected Devices, the framework will return an error object to application. This error object will have specific error code and error message. Based on these error code / error message, partner application can choose on what to inform the end user or retry any failed cases. A list of error codes is provided in the appendix.

2. Bluetooth Pairing Lost Scenarios

Partner applications can come across special cases when Omron Connected Devices does not pair or transfer with a smartphone.

a. Pairing Omron Connected Devices that will pair with flashing “P”:
When end user tries to pair such device in flash “o” for the first time, the Bluetooth profile for this device will get corrupted and end user will not be able to pair the device going forward. In this case, OmronConnectivityLibrary.framework will provide back below details:

Error code: 6022
Error message: Device authentication error
(Device is not in registered mode)

If partner application receives these details, it should inform end user to manually delete the Bluetooth profile for that Omron connected device from smartphone’s Bluetooth settings.

b. When end user has too many Bluetooth connected devices paired to smartphone, he/she might not be able to transfer data from an already connected Omron Connected Device with OmronConnectivityLibrary.framework

In this scenario partner application can retry transferring from Omron Connected Devices and if retries exceeds a maximum threshold, application can inform end user to remove the Bluetooth profile for that Omron Connected Device and re-pair the same device.

3. Bluetooth Data Transfer Scenarios

a. BLE broadcasting with unsend data:
   For Omron blood pressure monitors like BP786/CAN and BP761/CAN, it will broadcast till the unsend data is read.
   Other devices have a 60 minutes max timeout for BLE broadcasting. Once unsend data is read it will stop broadcasting.

4. Date and Time on Blood Pressure monitor

When date and time on your Omron Connected device is not set when the reading(s) are taken, readings transferred will show the date as "January, 01, 2015 12:01AM" or “January, 01, 2014 12:01AM”. Please make sure to set the date and time on your blood pressure monitor so it can record the correct time. Also if the date and time is not set according to the blood pressure monitor instructions, your readings may not transfer to the Partner app. In some cases, readings will be transferred with 0 date time and partner application need to handle them accordingly.

IX. Appendix: -

1. Error Codes (For Reference)

CODE	GENERAL DESCRIPTION
6001 – 6011 6021	Device Pairing error
6015	Bluetooth Not Supported
6016, 60161	Bluetooth Off, Bluetooth Unauthorized
6012 6017 – 6024 6031 6032	Device Connection Error
6025	Device Encryption Error
6029	Device Scan Timeout
6030	Connection Timeout
All Other Error Codes	Device Communication Error

Audio Error codes:

CODE	GENERAL DESCRIPTION
8193	Communication timeout
4353	Failed to instantiate the AudioQueue
4354	Failed to set sample rate in the AudioSession
4355	Failed to activate the AudioSession
4356	Failed to set the AudioSession category
4357	Failed to instantiate the AudioUnit
4358	Failed to enable the AudioUnit I/O
4359	Failed to initialize the AudioUnit
4360	Failed to enable the AudioUnit output stream format
4361	Failed to enable the AudioUnit input stream format
4362	Failed to set the AudioUnit callback
4363	AudioUnit output is failed
4364	Failed to render the AudioUnit
4365	The communication was interrupted because other application used the microphone.
4366	The audio recording permission is needed.

Suggested User Action:

ERROR	SUGGESTED USER ACTION
Device Pairing error	Your readings did not transfer, please make sure your device is within 10 feet of your blood pressure monitor and turned on, and then try again.
Bluetooth Not Supported	Your smartphone does not support Bluetooth
Bluetooth Off	It looks like Bluetooth is OFF. Please go to your smartphone Settings and turn Bluetooth ON.
Device Connection Error	**During Pairing:** Clear the screen by pressing the "Start/Stop" button. Press and hold the transfer button on your blood pressure monitor until you see "P" on your monitor. **During Transfer:** Your readings did not transfer, please make sure your device is within 10 feet of your blood pressure monitor and turned on, and then try again.
Device Scan Timeout	We can't find your device. Please make sure your device is within 10 feet of your blood pressure monitor and turned on, and then try again.
Connection Timeout	We can't find your device. Please make sure your device is within 10 feet of your blood pressure monitor and turned on, and then try again.
Device Communication Error	Your readings did not transfer, please make sure your device is within 10 feet of your blood pressure monitor and turned on, and then try again.
Device Encryption Error	Please remove the device from Bluetooth Settings in your smartphone and repair the device.

2. Sample Data from Library

a. Vital Data

BLOOD PRESSURE:

```
(
    {
    OMRONVitalDataArtifactDetectionKey = 0;
    OMRONVitalDataDiastolicKey = 79;
    OMRONVitalDataIHBDetectionKey = 0;
    OMRONVitalDataIrregularFlagKey = 0;
    OMRONVitalDataMeasurementDateKey = "2018-06-09 22:01:33 +0000";
    OMRONVitalDataMeasurementStartDateKey = 1528581693;
    OMRONVitalDataMovementFlagKey = 0;
    OMRONVitalDataPulseKey = 81;
    OMRONVitalDataSystolicKey = 116;
  },
    {
    OMRONVitalDataArtifactDetectionKey = 0;
    OMRONVitalDataDiastolicKey = 85;
    OMRONVitalDataIHBDetectionKey = 0;
    OMRONVitalDataIrregularFlagKey = 0;
    OMRONVitalDataMeasurementDateKey = "2018-06-11 16:08:15 +0000";
    OMRONVitalDataMeasurementStartDateKey = 1528733295;
    OMRONVitalDataMovementFlagKey = 0;
    OMRONVitalDataPulseKey = 98;
    OMRONVitalDataSystolicKey = 116;
  },
    {
    OMRONVitalDataArtifactDetectionKey = 0;
    OMRONVitalDataDiastolicKey = 80;
    OMRONVitalDataIHBDetectionKey = 0;
    OMRONVitalDataIrregularFlagKey = 0;
    OMRONVitalDataMeasurementDateKey = "2018-06-11 16:08:46 +0000";
    OMRONVitalDataMeasurementStartDateKey = 1528733326;
    OMRONVitalDataMovementFlagKey = 0;
    OMRONVitalDataPulseKey = 93;
    OMRONVitalDataSystolicKey = 118;
  }
```

ACTIVITY:

Aerobic Step Data

```
(
    {
    OMRONActivityDataDividedDataKey =   (
                {
            OMRONActivityDividedDataMeasurementDetailsKey =   (
                            (
                0,
                60,
                0
              )
            );
            OMRONActivityDividedDataMeasurementKey = 0;
            OMRONActivityDividedDataPeriodTimeKey = 60;
            OMRONActivityDividedDataStartDateKey = 1528675200;
        },
                {
            OMRONActivityDividedDataMeasurementDetailsKey =    (
                            (
                0,
                60,
                0
              )
            );
            OMRONActivityDividedDataMeasurementKey = 0;
            OMRONActivityDividedDataPeriodTimeKey = 60;
            OMRONActivityDividedDataStartDateKey = 1528678800;
        },
                            .
                            .
                            .
                {
            OMRONActivityDividedDataMeasurementDetailsKey =   (
                            (
                0,
                60,
                0
              )
            );
            OMRONActivityDividedDataMeasurementKey = 0;
            OMRONActivityDividedDataPeriodTimeKey = 60;
            OMRONActivityDividedDataStartDateKey = 1528758000;
        }
    );
    OMRONActivityDataEndDateKey = 1528675200;
    OMRONActivityDataMeasurementKey = 0;
    OMRONActivityDataSequenceKey = 120;
    OMRONActivityDataStartDateKey = 1528675200;
  }
)
```

Step Data

```
(
    {
    OMRONActivityDataDividedDataKey =  (
                {
            OMRONActivityDividedDataMeasurementDetailsKey =  (
                            (
                0,
                60,
                0
              )
            );
            OMRONActivityDividedDataMeasurementKey = 0;
            OMRONActivityDividedDataPeriodTimeKey = 60;
            OMRONActivityDividedDataStartDateKey = 1528675200;
          },
                                    .
                                    .
                                    .
                                     {
            OMRONActivityDividedDataMeasurementDetailsKey =   (
                            (
                0,
                60,
                873
              )
            );
            OMRONActivityDividedDataMeasurementKey = 873;
            OMRONActivityDividedDataPeriodTimeKey = 60;
            OMRONActivityDividedDataStartDateKey = 1528714800;
          },
                .
                            .
                            .

                                    {
            OMRONActivityDividedDataMeasurementDetailsKey =   (
                            (
                0,
                60,
                0
              )
            );
            OMRONActivityDividedDataMeasurementKey = 0;
            OMRONActivityDividedDataPeriodTimeKey = 60;
            OMRONActivityDividedDataStartDateKey = 1528758000;
          }
    );
    OMRONActivityDataEndDateKey = 1528675200;
    OMRONActivityDataMeasurementKey = 873;
    OMRONActivityDataSequenceKey = 120;
    OMRONActivityDataStartDateKey = 1528675200;
  }
)
```

Distance Data

```
(
    {
    OMRONActivityDataDividedDataKey =  (
                {
            OMRONActivityDividedDataMeasurementDetailsKey =   (
                            (
                0,
                60,
                0
              )
            );
            OMRONActivityDividedDataMeasurementKey = 0;
            OMRONActivityDividedDataPeriodTimeKey = 60;
            OMRONActivityDividedDataStartDateKey = 1528675200;
          },
                                    .
                                    .
                                    .
                                    {
            OMRONActivityDividedDataMeasurementDetailsKey =   (
                            (
                0,
                60,
                3
              )
            );
            OMRONActivityDividedDataMeasurementKey = "0.3";
            OMRONActivityDividedDataPeriodTimeKey = 60;
            OMRONActivityDividedDataStartDateKey = 1528714800;
          },

                                    .
                                    .
                                    .
                {
            OMRONActivityDividedDataMeasurementDetailsKey =   (
                            (
                0,
                60,
                0
              )
            );
            OMRONActivityDividedDataMeasurementKey = 0;
            OMRONActivityDividedDataPeriodTimeKey = 60;
            OMRONActivityDividedDataStartDateKey = 1528758000;
          }
    );
    OMRONActivityDataEndDateKey = 1528675200;
    OMRONActivityDataMeasurementKey = "0.3";
    OMRONActivityDataSequenceKey = 120;
    OMRONActivityDataStartDateKey = 1528675200;
  }
)
```

Calories Data

```
(
    {
    OMRONActivityDataDividedDataKey =  (
                {
            OMRONActivityDividedDataMeasurementDetailsKey =   (
                            (
                0,
                60,
                0
              )
            );
            OMRONActivityDividedDataMeasurementKey = 0;
            OMRONActivityDividedDataPeriodTimeKey = 60;
            OMRONActivityDividedDataStartDateKey = 1528675200;
        },
                                    .
                                    .
                                    .
                                    {
            OMRONActivityDividedDataMeasurementDetailsKey =    (
                            (
                0,
                60,
                33
              )
            );
            OMRONActivityDividedDataMeasurementKey = 33;
            OMRONActivityDividedDataPeriodTimeKey = 60;
            OMRONActivityDividedDataStartDateKey = 1528714800;
        },
                                    .
                                    .
                                    .
                                                {
            OMRONActivityDividedDataMeasurementDetailsKey =  (
                            (
                0,
                60,
                0
              )
            );
            OMRONActivityDividedDataMeasurementKey = 0;
            OMRONActivityDividedDataPeriodTimeKey = 60;
            OMRONActivityDividedDataStartDateKey = 1528758000;
        }
    );
    OMRONActivityDataEndDateKey = 1528675200;
    OMRONActivityDataMeasurementKey = 33;
    OMRONActivityDataSequenceKey = 120;
    OMRONActivityDataStartDateKey = 1528675200;
  }
)
```

SLEEP:

```
(
    {
    OMRONSleepArousalDuringSleepTimeKey = 0;
    OMRONSleepBodyMotionLevelKey =  (
                (
            0,
            5,
            1
        ),
                (
            5,
            5,
            1
        ),
                (
            10,
            5,
            1
        )
    );
    OMRONSleepDataEndDateKey = 1528490040;
    OMRONSleepDataStartDateKey = 1528489260;
    OMRONSleepSleepEfficiencyKey = 100;
    OMRONSleepSleepOnsetTimeKey = 1528489260;
    OMRONSleepTimeInBedKey = 1528489260;
    OMRONSleepTotalSleepTimeKey = 13;
    OMRONSleepWakeTimeKey = 1528490040;
  },
    {
    OMRONSleepArousalDuringSleepTimeKey = 0;
    OMRONSleepBodyMotionLevelKey =  (
                (
            0,
            5,
            2
        ),
                (
            5,
            5,
            2
        )
    );
    OMRONSleepDataEndDateKey = 1528734480;
    OMRONSleepDataStartDateKey = 1528734000;
    OMRONSleepSleepEfficiencyKey = 0;
    OMRONSleepSleepOnsetTimeKey = 0;
    OMRONSleepTimeInBedKey = 1528734000;
    OMRONSleepTotalSleepTimeKey = 0;
    OMRONSleepWakeTimeKey = 1528734480;
  },
    {
    OMRONSleepArousalDuringSleepTimeKey = 5;
    OMRONSleepBodyMotionLevelKey =  (
                (
            0,
            5,
            2
        ),
```

```
                (
            5,
            5,
            1
        ),
                (
            10,
            5,
            1
        ),
                (
            15,
            5,
            0
        ),
                (
            20,
            5,
            0
        ),
                (
            25,
            5,
            0
        ),
                (
            30,
            5,
            0
        ),
                (
            35,
            5,
            0
        ),
                (
            40,
            5,
            2
        )
    );
    OMRONSleepDataEndDateKey = 1528736700;
    OMRONSleepDataStartDateKey = 1528734480;
    OMRONSleepSleepEfficiencyKey = "83.7";
    OMRONSleepSleepOnsetTimeKey = 1528734540;
    OMRONSleepTimeInBedKey = 1528734480;
    OMRONSleepTotalSleepTimeKey = 31;
    OMRONSleepWakeTimeKey = 1528736700;
  }
)
```

RECORD:

```
(
  {
    OMRONRecordDataDateKey = 1528734139;
  },
  {
    OMRONRecordDataDateKey = 1528734319;
  }
)
```

WEIGHT:

```
{
    OMRONVitalDataWeightKey = (
                {
            OMRONWeightBMIKey = "28.2";
            OMRONWeightBodyFatLevelClassificationKey = 9;
            OMRONWeightBodyFatPercentageKey = "24.9";
            OMRONWeightDataSequenceKey = 7;
            OMRONWeightDataStartDateKey = 1554899687;
            OMRONWeightDataUserIdKey = 2;
            OMRONWeightKey = "74.55";
            OMRONWeightRestingMetabolismKey = 1549;
            OMRONWeightSkeletalMuscleLevelClassificationKey = 3;
            OMRONWeightSkeletalMusclePercentageKey = "31.1";
            OMRONWeightVisceralFatLevelClassificationKey = 9;
            OMRONWeightVisceralFatLevelKey = 10;
        }
    );
}
```

WHEEZE:

```
{
    OMRONVitalDataWheezeKey = (
                {
            OMRONWheezeDataSequenceKey = 1;
            OMRONWheezeDataStartDateKey = 1616903435;
            OMRONWheezeDataUserIdKey = 1;
            OMRONWheezeErrorDecreaseBreathingSoundLevelKey = 0;
            OMRONWheezeErrorNoiseKey = 0;
            OMRONWheezeErrorSurroundingNoiseKey = 0;
            OMRONWheezeKey = 0;
        },
                {
            OMRONWheezeDataSequenceKey = 2;
            OMRONWheezeDataStartDateKey = 1616903494;
            OMRONWheezeDataUserIdKey = 1;
            OMRONWheezeErrorDecreaseBreathingSoundLevelKey = 0;
            OMRONWheezeErrorNoiseKey = 0;
            OMRONWheezeErrorSurroundingNoiseKey = 0;
            OMRONWheezeKey = 1;
        },
                {
            OMRONWheezeDataSequenceKey = 3;
            OMRONWheezeDataStartDateKey = 1616903567;
            OMRONWheezeDataUserIdKey = 1;
            OMRONWheezeErrorDecreaseBreathingSoundLevelKey = 1;
            OMRONWheezeErrorNoiseKey = 0;
            OMRONWheezeErrorSurroundingNoiseKey = 0;
            OMRONWheezeKey = 2;
        },
                {
            OMRONWheezeDataSequenceKey = 13;
            OMRONWheezeDataStartDateKey = 1619282042;
            OMRONWheezeDataUserIdKey = 1;
            OMRONWheezeErrorDecreaseBreathingSoundLevelKey = 0;
            OMRONWheezeErrorNoiseKey = 0;
            OMRONWheezeErrorSurroundingNoiseKey = 1;
            OMRONWheezeKey = 2;
        }
    );
}
```

PULSE OXIMETER:

```
{
    OMRONVitalDataPulseOximeterKey = (
                {
            OMRONPulseOximeterDataSequenceKey = 0;
            OMRONPulseOximeterDataStartDateKey = 1627341696;
            OMRONPulseOximeterDataUserIdKey = 1;
            OMRONPulseOximeterMeasurementSupportFieldKey = 32;
            OMRONPulseOximeterPulseRateKey = 85;
            OMRONPulseOximeterSPO2LevelKey = 98;
            OMRONPulseOximeterSensorStatusKey = 32;
        },
                {
            OMRONPulseOximeterDataSequenceKey = 0;
            OMRONPulseOximeterDataStartDateKey = 1627341656;
            OMRONPulseOximeterDataUserIdKey = 1;
            OMRONPulseOximeterMeasurementSupportFieldKey = 544;
            OMRONPulseOximeterPulseRateKey = 87;
            OMRONPulseOximeterSPO2LevelKey = 98;
            OMRONPulseOximeterSensorStatusKey = 32;
        },
                {
            OMRONPulseOximeterDataSequenceKey = 0;
            OMRONPulseOximeterDataStartDateKey = 1627341675;
            OMRONPulseOximeterDataUserIdKey = 1;
            OMRONPulseOximeterMeasurementSupportFieldKey = 544;
            OMRONPulseOximeterPulseRateKey = 82;
            OMRONPulseOximeterSPO2LevelKey = 98;
            OMRONPulseOximeterSensorStatusKey = 32;
        }
    );
}
```

TEMPERATURE:

```
{
         OMRONVitalDataTemperatureKey = (
                     {
                OMRONTemperatureCelsiusKey = "33.45500183105469";
                OMRONTemperatureDataSequenceKey = 24;
                OMRONTemperatureDataStartDateKey = 1621205609;
                OMRONTemperatureDeviceModelKey = 1;
                OMRONTemperatureKey = "92.2";
                OMRONTemperatureUnitKey = 0;
            }
         );
     }

 {
        OMRONVitalDataTemperatureKey = (
                    {
               OMRONTemperatureCelsiusKey = 65535;
               OMRONTemperatureDataSequenceKey = 30;
               OMRONTemperatureDataStartDateKey = 1621702061;
               OMRONTemperatureDeviceModelKey = 1;
               OMRONTemperatureKey = 65535;
               OMRONTemperatureLevelKey = 1;
               OMRONTemperatureUnitKey = 0;
            }
        );
    }
```

b. Device Information

```
{
    OMRONDeviceInformationDisplayNameKey = "BCM-500";
    OMRONDeviceInformationIdentityNameKey = "HBF-222T_Z";
    OMRONDeviceInformationLocalNameKey = "BLEsmart_00010208EC21E5794858";
    OMRONDeviceInformationSerialIdKey = 584879feffe521ec;
    OMRONDeviceInformationUUIDKey = "D116915B-A839-AAB1-3D49-A411A3AEA1DA";
    OMRONDeviceInformationiBeaconMajorValueKey = 1;
    OMRONDeviceInformationiBeaconMinorValueKey = 1;
    OMRONDeviceInformationiBeaconProximityUUIDKey = "8D696C92-6A22-43F7-9A5E-170DFBCFB630";
}
```

c. Device Settings

I. Device Settings

Blood Pressure:

a) (User 1 registered and User 2 is available)

```
{
        OMRONDevicePersonalSettingsKey = {
            1 = {
                OMRONDevicePersonalSettingsUserDateOfBirthKey = 19000101;
            };
            2 = {
                OMRONDevicePersonalSettingsUserDateOfBirthKey = 19000101;
            };
        };
    },
        {
        OMRONDeviceSettingsKey = {
            OMRONDeviceSettingsAvailableUsersKey = (
                1,
                2
            );
            OMRONDeviceSettingsRegisteredUsersKey = (
                1
            );
        };
    }
```

b) (Both Users 1 and 2 are registered)

```
{
        OMRONDevicePersonalSettingsKey = {
            1 = {
                OMRONDevicePersonalSettingsUserDateOfBirthKey = 19000101;
            };
            2 = {
                OMRONDevicePersonalSettingsUserDateOfBirthKey = 19000101;
            };
        };
    },
        {
        OMRONDeviceSettingsKey = {
            OMRONDeviceSettingsAvailableUsersKey = (
                1,
                2
            );
            OMRONDeviceSettingsRegisteredUsersKey = (
                1,
                2
            );
        };
    }
```

Activity:

```
{
        OMRONDeviceSleepSettingsKey = {
            OMRONDeviceSleepSettingsAutomaticKey = 1;
            OMRONDeviceSleepSettingsAutomaticStartTimeKey = 21;
            OMRONDeviceSleepSettingsAutomaticStopTimeKey = 7;
        };
    },
        {
        OMRONDeviceTimeSettingsKey = {
            OMRONDeviceTimeSettingsFormatKey = 1;
        };
    },
        {
        OMRONDeviceWeightSettingsKey = {
            OMRONDeviceWeightSettingsUnitKey = 0;
        };
    },
        {
        OMRONDeviceDistanceSettingsKey = {
            OMRONDeviceDistanceSettingsUnitKey = 1;
        };
    },
        {
        OMRONDeviceDateSettingsKey = {
            OMRONDeviceDateSettingsFormatKey = 0;
        };
    },
{
        OMRONDeviceAlarmSettingsKey = (
                        {
                OMRONDeviceAlarmSettingsDaysKey = {
                    OMRONDeviceAlarmSettingsDayFridayKey = 0;
                    OMRONDeviceAlarmSettingsDayMondayKey = 0;
                    OMRONDeviceAlarmSettingsDaySaturdayKey = 0;
                    OMRONDeviceAlarmSettingsDaySundayKey = 0;
                    OMRONDeviceAlarmSettingsDayThursdayKey = 0;
                    OMRONDeviceAlarmSettingsDayTuesdayKey = 0;
                    OMRONDeviceAlarmSettingsDayWednesdayKey = 0;
                };
                OMRONDeviceAlarmSettingsTimeKey = {
                    OMRONDeviceAlarmSettingsHourKey = 0;
                    OMRONDeviceAlarmSettingsMinuteKey = 0;
                };
                OMRONDeviceAlarmSettingsTypeKey = 3;
            },
                        {
                OMRONDeviceAlarmSettingsDaysKey = {
                    OMRONDeviceAlarmSettingsDayFridayKey = 0;
                    OMRONDeviceAlarmSettingsDayMondayKey = 0;
                    OMRONDeviceAlarmSettingsDaySaturdayKey = 0;
                    OMRONDeviceAlarmSettingsDaySundayKey = 0;
                    OMRONDeviceAlarmSettingsDayThursdayKey = 0;
                    OMRONDeviceAlarmSettingsDayTuesdayKey = 0;
                    OMRONDeviceAlarmSettingsDayWednesdayKey = 0;
                };
                OMRONDeviceAlarmSettingsTimeKey = {
                    OMRONDeviceAlarmSettingsHourKey = 0;
                    OMRONDeviceAlarmSettingsMinuteKey = 0;
                };
                OMRONDeviceAlarmSettingsTypeKey = 3;
            },
```

```
                    {
                OMRONDeviceAlarmSettingsDaysKey = {
                    OMRONDeviceAlarmSettingsDayFridayKey = 0;
                    OMRONDeviceAlarmSettingsDayMondayKey = 0;
                    OMRONDeviceAlarmSettingsDaySaturdayKey = 0;
                    OMRONDeviceAlarmSettingsDaySundayKey = 0;
                    OMRONDeviceAlarmSettingsDayThursdayKey = 0;
                    OMRONDeviceAlarmSettingsDayTuesdayKey = 0;
                    OMRONDeviceAlarmSettingsDayWednesdayKey = 0;
                };
                OMRONDeviceAlarmSettingsTimeKey = {
                    OMRONDeviceAlarmSettingsHourKey = 0;
                    OMRONDeviceAlarmSettingsMinuteKey = 0;
                };
                OMRONDeviceAlarmSettingsTypeKey = 3;
            },
                    {
                OMRONDeviceAlarmSettingsDaysKey = {
                    OMRONDeviceAlarmSettingsDayFridayKey = 0;
                    OMRONDeviceAlarmSettingsDayMondayKey = 0;
                    OMRONDeviceAlarmSettingsDaySaturdayKey = 0;
                    OMRONDeviceAlarmSettingsDaySundayKey = 0;
                    OMRONDeviceAlarmSettingsDayThursdayKey = 0;
                    OMRONDeviceAlarmSettingsDayTuesdayKey = 0;
                    OMRONDeviceAlarmSettingsDayWednesdayKey = 0;
                };
                OMRONDeviceAlarmSettingsTimeKey = {
                    OMRONDeviceAlarmSettingsHourKey = 0;
                    OMRONDeviceAlarmSettingsMinuteKey = 0;
                };
                OMRONDeviceAlarmSettingsTypeKey = 3;
            },
                    {
                OMRONDeviceAlarmSettingsDaysKey = {
                    OMRONDeviceAlarmSettingsDayFridayKey = 0;
                    OMRONDeviceAlarmSettingsDayMondayKey = 0;
                    OMRONDeviceAlarmSettingsDaySaturdayKey = 0;
                    OMRONDeviceAlarmSettingsDaySundayKey = 0;
                    OMRONDeviceAlarmSettingsDayThursdayKey = 0;
                    OMRONDeviceAlarmSettingsDayTuesdayKey = 0;
                    OMRONDeviceAlarmSettingsDayWednesdayKey = 0;
                };
                OMRONDeviceAlarmSettingsTimeKey = {
                    OMRONDeviceAlarmSettingsHourKey = 0;
                    OMRONDeviceAlarmSettingsMinuteKey = 0;
                };
                OMRONDeviceAlarmSettingsTypeKey = 3;
            }
        );
    },
        {
        OMRONDevicePersonalSettingsKey = {
            1 = {
                OMRONDevicePersonalSettingsTargetSleepKey = 420;
                OMRONDevicePersonalSettingsTargetStepsKey = 1000;
                OMRONDevicePersonalSettingsUserHeightKey = 18050;
                OMRONDevicePersonalSettingsUserWeightKey = 9435;
            };
        };
    },
        {
        OMRONDeviceSettingsKey = {
            OMRONDeviceSettingsAvailableUsersKey = (
                1
            );
            OMRONDeviceSettingsRegisteredUsersKey = (
                1
            );
        };
    }
```

Body Composition:

```
(
        {
        OMRONDeviceTimeSettingsKey = {
            OMRONDeviceTimeSettingsFormatKey = 0;
        };
    },
        {
        OMRONDeviceWeightSettingsKey = {
            OMRONDeviceWeightSettingsUnitKey = 1;
        };
    },
        {
        OMRONDeviceDateSettingsKey = {
            OMRONDeviceDateSettingsFormatKey = 1;
        };
    },
        {
        OMRONDevicePersonalSettingsKey = {
            1 = {
                OMRONDevicePersonalSettingsUserDateOfBirthKey = 19920202;
                OMRONDevicePersonalSettingsUserGenderKey = 1;
                OMRONDevicePersonalSettingsUserHeightKey = 1625;
                OMRONDevicePersonalSettingsWeightKey = {
                    OMRONDevicePersonalSettingsWeightDCIKey = 5;
                    OMRONDevicePersonalSettingsWeightDisplayEnableBMIKey = 1;
                    OMRONDevicePersonalSettingsWeightDisplayEnableBodyFatKey = 1;
                    OMRONDevicePersonalSettingsWeightDisplayEnableRestingMetabolismKey = 1;
                    OMRONDevicePersonalSettingsWeightDisplayEnableSkeletalMuscleLevelKey = 1;
                    OMRONDevicePersonalSettingsWeightDisplayEnableVisceralFatLevelKey = 1;
                    OMRONDevicePersonalSettingsWeightDisplayPriorityBMIKey = 6;
                    OMRONDevicePersonalSettingsWeightDisplayPriorityBodyFatKey = 1;
                    OMRONDevicePersonalSettingsWeightDisplayPriorityRestingMetabolismKey = 5;
                    OMRONDevicePersonalSettingsWeightDisplayPrioritySkeletalMuscleLevelKey = 3;
                    OMRONDevicePersonalSettingsWeightDisplayPriorityVisceralFatLevelKey = 2;
                };
            };
            2 = {
                OMRONDevicePersonalSettingsUserDateOfBirthKey = 19860606;
                OMRONDevicePersonalSettingsUserGenderKey = 1;
                OMRONDevicePersonalSettingsUserHeightKey = 1625;
                OMRONDevicePersonalSettingsWeightKey = {
                    OMRONDevicePersonalSettingsWeightDCIKey = 57;
                    OMRONDevicePersonalSettingsWeightDisplayEnableBMIKey = 1;
                    OMRONDevicePersonalSettingsWeightDisplayEnableBodyFatKey = 1;
                    OMRONDevicePersonalSettingsWeightDisplayEnableRestingMetabolismKey = 1;
                    OMRONDevicePersonalSettingsWeightDisplayEnableSkeletalMuscleLevelKey = 1;
                    OMRONDevicePersonalSettingsWeightDisplayEnableVisceralFatLevelKey = 1;
                    OMRONDevicePersonalSettingsWeightDisplayPriorityBMIKey = 6;
                    OMRONDevicePersonalSettingsWeightDisplayPriorityBodyFatKey = 1;
                    OMRONDevicePersonalSettingsWeightDisplayPriorityRestingMetabolismKey = 5;
                    OMRONDevicePersonalSettingsWeightDisplayPrioritySkeletalMuscleLevelKey = 3;
                    OMRONDevicePersonalSettingsWeightDisplayPriorityVisceralFatLevelKey = 2;
                };
            };
```

```
3 = {
                OMRONDevicePersonalSettingsUserDateOfBirthKey = 19860606;
                OMRONDevicePersonalSettingsUserGenderKey = 1;
                OMRONDevicePersonalSettingsUserHeightKey = 1625;
                OMRONDevicePersonalSettingsWeightKey = {
                   OMRONDevicePersonalSettingsWeightDCIKey = 8;
                   OMRONDevicePersonalSettingsWeightDisplayEnableBMIKey = 1;
                   OMRONDevicePersonalSettingsWeightDisplayEnableBodyFatKey = 1;
                   OMRONDevicePersonalSettingsWeightDisplayEnableRestingMetabolismKey = 1;
                   OMRONDevicePersonalSettingsWeightDisplayEnableSkeletalMuscleLevelKey = 1;
                   OMRONDevicePersonalSettingsWeightDisplayEnableVisceralFatLevelKey = 1;
                   OMRONDevicePersonalSettingsWeightDisplayPriorityBMIKey = 6;
                   OMRONDevicePersonalSettingsWeightDisplayPriorityBodyFatKey = 1;
                   OMRONDevicePersonalSettingsWeightDisplayPriorityRestingMetabolismKey = 5;
                   OMRONDevicePersonalSettingsWeightDisplayPrioritySkeletalMuscleLevelKey = 3;
                   OMRONDevicePersonalSettingsWeightDisplayPriorityVisceralFatLevelKey = 2;
                };
            };
            4 = {
                OMRONDevicePersonalSettingsUserDateOfBirthKey = 19860606;
                OMRONDevicePersonalSettingsUserGenderKey = 1;
                OMRONDevicePersonalSettingsUserHeightKey = 1625;
                OMRONDevicePersonalSettingsWeightKey =                 {
                   OMRONDevicePersonalSettingsWeightDCIKey = 6;
                   OMRONDevicePersonalSettingsWeightDisplayEnableBMIKey = 1;
                   OMRONDevicePersonalSettingsWeightDisplayEnableBodyFatKey = 1;
                   OMRONDevicePersonalSettingsWeightDisplayEnableRestingMetabolismKey = 1;
                   OMRONDevicePersonalSettingsWeightDisplayEnableSkeletalMuscleLevelKey = 1;
                   OMRONDevicePersonalSettingsWeightDisplayEnableVisceralFatLevelKey = 1;
                   OMRONDevicePersonalSettingsWeightDisplayPriorityBMIKey = 6;
                   OMRONDevicePersonalSettingsWeightDisplayPriorityBodyFatKey = 1;
                   OMRONDevicePersonalSettingsWeightDisplayPriorityRestingMetabolismKey = 5;
                   OMRONDevicePersonalSettingsWeightDisplayPrioritySkeletalMuscleLevelKey = 3;
                   OMRONDevicePersonalSettingsWeightDisplayPriorityVisceralFatLevelKey = 2;
                };
            };
        };
    },
        {
        OMRONDeviceSettingsKey = {
            OMRONDeviceSettingsAvailableUsersKey = (
                3,
                1,
                4,
                2
            );
            OMRONDeviceSettingsRegisteredUsersKey = (
                3,
                1,
                4,
                2
            );
        };
    }
)
```

II. Device settings with user:

Blood Pressure:

```
{
    OMRONDevicePersonalSettingsUserDateOfBirthKey = 19000101;
}
```

Activity:

```
{
    OMRONDevicePersonalSettingsTargetSleepKey = 420;
    OMRONDevicePersonalSettingsTargetStepsKey = 1000;
    OMRONDevicePersonalSettingsUserHeightKey = 18050;
    OMRONDevicePersonalSettingsUserWeightKey = 9435;
}
```

Body Composition:

```
{
                OMRONDevicePersonalSettingsUserDateOfBirthKey = 19860606;
                OMRONDevicePersonalSettingsUserGenderKey = 1;
                OMRONDevicePersonalSettingsUserHeightKey = 1625;
                OMRONDevicePersonalSettingsWeightKey = {
                 OMRONDevicePersonalSettingsWeightDCIKey = 8;
                 OMRONDevicePersonalSettingsWeightDisplayEnableBMIKey = 1;
                 OMRONDevicePersonalSettingsWeightDisplayEnableBodyFatKey = 1;
                 OMRONDevicePersonalSettingsWeightDisplayEnableRestingMetabolismKey = 1;
                 OMRONDevicePersonalSettingsWeightDisplayEnableSkeletalMuscleLevelKey = 1;
                 OMRONDevicePersonalSettingsWeightDisplayEnableVisceralFatLevelKey = 1;
                 OMRONDevicePersonalSettingsWeightDisplayPriorityBMIKey = 6;
                 OMRONDevicePersonalSettingsWeightDisplayPriorityBodyFatKey = 1;
                 OMRONDevicePersonalSettingsWeightDisplayPriorityRestingMetabolismKey = 5;
                 OMRONDevicePersonalSettingsWeightDisplayPrioritySkeletalMuscleLevelKey = 3;
                 OMRONDevicePersonalSettingsWeightDisplayPriorityVisceralFatLevelKey = 2;
                };
};
```