OmronConnectivityLibrary Android

A Guide to integrate OmronConnectivityLibrary Framework with Third Party Apps

V 4.1

Revision History

Issue Date	Version	Details
06/01/2017	1.0	• Initial OmronConnectivityLibrary Integration Details
02/14/2018	1.1	• Updated V (2) to include new notification functionality to share device configuration status from library • Updated V (6) for new feature of pause and resume data transfer
02/21/2018	1.2	• Updated configuration for BP786/CAN and BP761/CAN
02/27/2018	1.3	• Additional device properties exposed through library • Updated VI (a) to include helper function to retrieve device configuration
04/05/2018	1.4	• Updated V (3) to add details on User Hash Id property
06/01/2018	1.5	• 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) and V(3) • 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
06/20/2018	1.6	• Updating library version and Partner code verification
07/02/2018	1.7	• Added support for Omron device BP7900
09/04/2018	1.8	• Add Google Analytics Integration to connectivity • Update Section IV (3) for Google Analytics instructions
09/11/2018	1.9	• Add support for BP300
09/14/2018	2.0	• Update Google Analytic to track additional details
12/06/2018	2.1	• Update device model name for BP7900
01/08/2019	2.2	• Add support for Omron device M700 Intelli IT
01/21/2019	2.3	• Add support for Omron device BP8000 (HeartGuide)
02/20/2019	2.4	• Add support for Omron device BP6350, BP4350, BP7250, BP7450 • Add new keys available in blood pressure in section V (7)

04/12/2019	2.5	• Added support for Omron Device HBF-222T_Z • Updated Section V (3)(d)(I) to add new keys under personal settings • Updated Section V (3) (d) to capture sleep settings for Activity • Added new methods in Section V (5) for connecting Omron connected devices • Added new methods in Section V (9) under update device settings • Added new Section V (10) (b) for device settings • Added new Appendix VIII (3) (c) for device settings in sample output data from library
05/01/2019	2.6	• Add support for Omron device EVOLV, RS7 Intelli IT, VIVA
06/04/2019	2.7	• Add support for Omron device BP5250, BP5350, BP5450
11/04/2019	2.8	• Add support for Omron device BP7250CAN, BP7350CAN, BP7450CAN
11/10/2019	2.9	• Add support for RS3 Intelli IT, HeartGuide, MIT5s, M4 Intelli IT, M7 Intelli IT 2nd Gen
03/03/2020	3.0	• Improvements in connectivity • Remove dependency for Google Analytics
03/31/2020	3.1	• Improvements in connectivity
06/18/2020	3.2	• 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	3.3	• Update asset catalog • Additional feature support for connectivity pairing and data transfer
09/15/2020	3.4	• Support for NightView (HEM-9601T) • Improvements to connectivity functionalities
01/05/2021	3.5	• Update device support
03/16/2021	3.6	• Updated to support Android Target API level 30 • Add support for device image asset catalog • Update device support
07/15/2021	3.7	• Improvements to connectivity functionalities • Update for Blood pressure measurement mode flag
09/10/2021	3.8	• Add support for HWZ-1000T-E, MC-280B, HPO-300T, HN-300T2, HEM-7530T-E3

		• Improvements to connectivity functionalities
10/15/2021	3.9	• Add support for SC-150, M2 Intelli IT, X2 Smart, M300 Intelli IT • Improvements to connectivity functionalities
11/05/2021	4.0	• M2 Intelli IT support fix • Improvements to connectivity functionalities
16/02/2022	4.1	• Improvements to connectivity functionalities

Table of Contents

Getting Started

I. Introduction

The OmronConnectivityLibrary allows Omron Partner Android applications to interact with Omron Connected Devices. This allows Omron partners to retrieve vital data from Omron Connected devices to Android device. Partners can use the data to build feature-rich user experience. This document will guide you to add the OmronConnectivityLibrary to an Android 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

- ➢ Android Studio
- ➢ Target and Compile Android SDK 30 or higher.
- ➢ Min SDK 18 or higher

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/

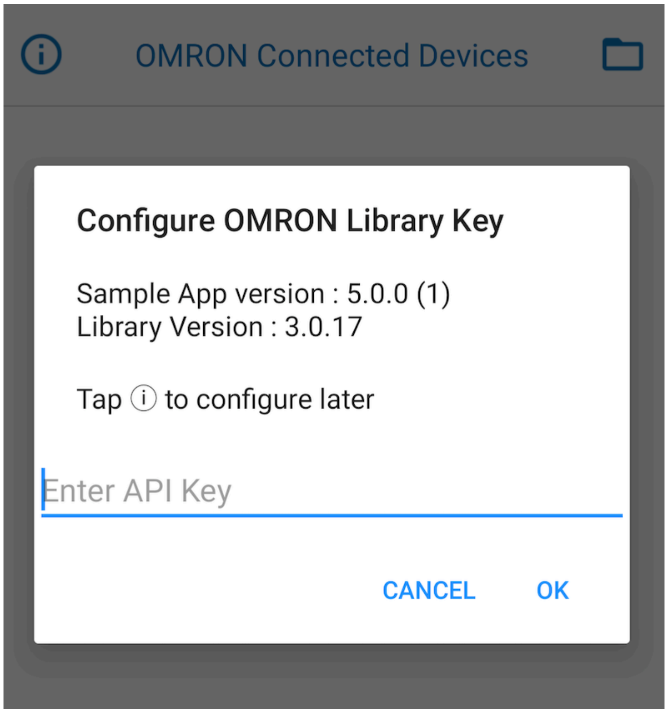
IV. Running the Sample Application

1. Open the sample app

In Android Studio go to File > Open and select the OmronConnectivtySample application. Generate build and 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.



Configure OMRON Library Key

Sample App version : 5.0.0 (1)
Library Version : 3.0.17

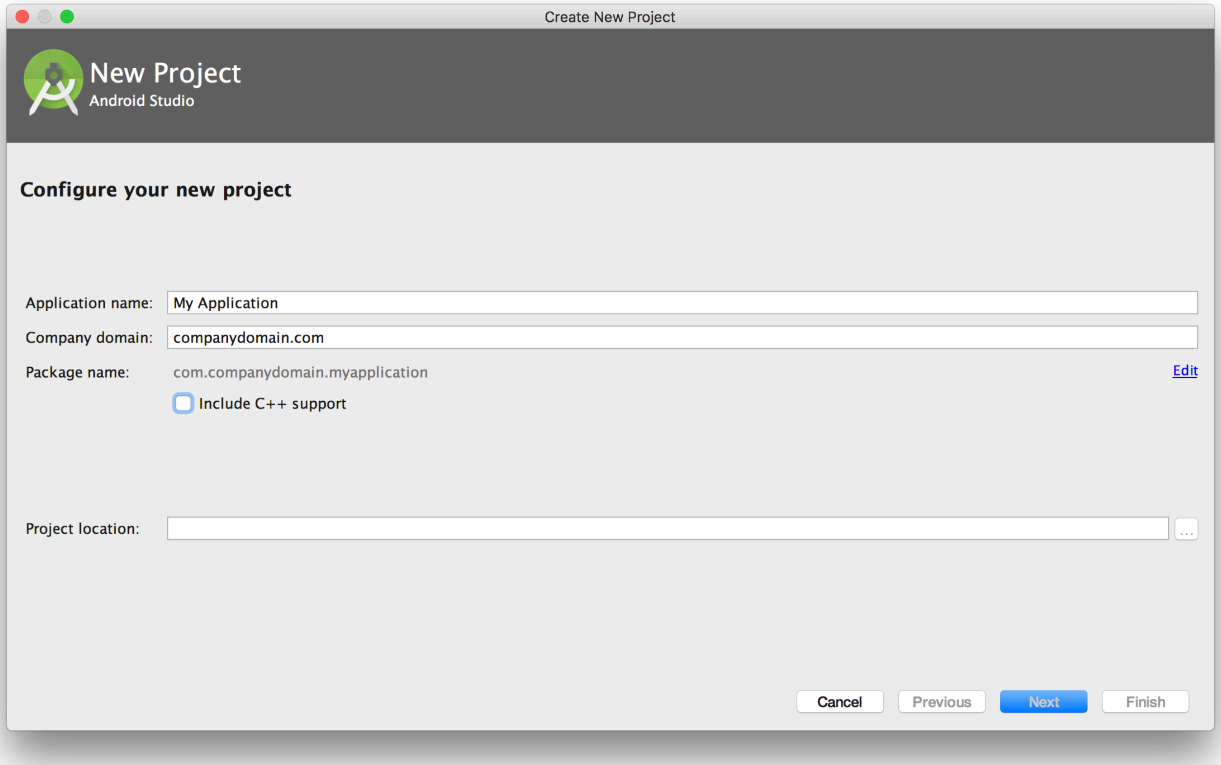
Tap ⓘ to configure later

Enter API Key

V. Configuring Project for OmronConnectivityLibrary

1. Create a new Android Studio Project

In Android Studio go to File > New > New Project and create a new project. Select appropriate Min SDK following library pre-requisites.



Configure your new project

Application name: My Application

Company domain: companydomain.com

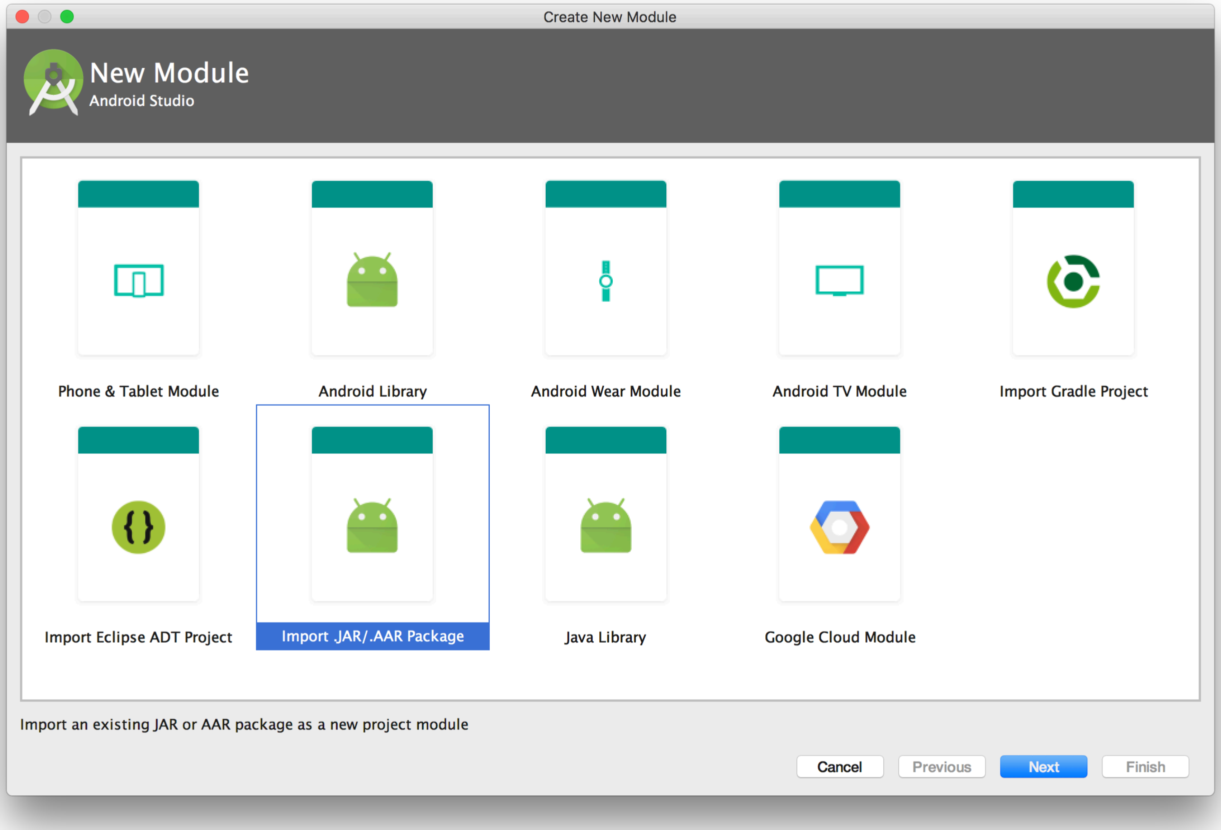
Package name: com.companydomain.myapplication

□ Include C++ support

Project location:

2. Add OmronConnectivityLibrary AAR

OmronConnectivityLibrary.aar is required to connect to Omron Connected Devices. In order to add the library AAR file to the project, it has to be imported as a new project module. Please go to File > New > New Module and select Import .JAR/.AAR Package option. Now select OmronConnectivityLibrary.aar file and complete the process.



Import an existing JAR or AAR package as a new project module

3. Adding OmronConnectivityLibraryAssets AAR (optional)

OmronConnectivityLibraryAssets is required to get the image and thumbnails of Omron Connected Devices. Following above step, add the aar file to project. Using this will increase the size of your application package.

VI. OmronConnectivityLibrary Integration

1. Initializing OmronConnectivityLibrary

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 is used to validate the authenticity of Partner.

This is a mandatory step to start using Omron Connectivity Library. Typically this is done within Application class.

```
OmronPeripheralManager.sharedManager(this).setAPIKey("PARTNER_API_KEY", null);
```

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

CODE	DESCRIPTION
OMRONConfigurationStatus.OMRONConfigurationPartnerAuthenticationError	Partner is not authorized

Below permissions are required by the library and is added to library configuration. This will get merged with your application's Android Manifest file.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.FINE_LOCATION" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

Bluetooth permission is required to perform any Bluetooth communication, such as requesting or accepting a connection and transferring data. Bluetooth admin permission is required, in addition, for the ability to discover Bluetooth devices.

Since Android 6.0, `ACCESS FINE LOCATION` or `ACCESS COARSE LOCATION` permission is required to identify near by external Bluetooth devices. *Reference:https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html#behavior-hardware-id*

Disclaimer : For latest changes w.r.t location services for Bluetooth compatibility, partner applications need to follow Google Android guidelines to use location permissions.

Additionally, if audio devices are used, then audio record permission need to be granted.

```
<uses-permission android:name="android.permission.RECORD_AUDIO " />
```

2. Retrieve Configurations

Omron Device configurations need to be retrieved to identify different Omron Connected Devices. 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(this).retrieveManagerConfiguration(this).get(OmronConstants.
OMRONBLEConfigDeviceKey)
```

This function returns a dictionary of configuration, in which Omron Connected Devices are available in one of the key. OmronConnectivityLibrary uses key OMRONBLEConfigDeviceKey to retrieve Omron Devices list. This is a list of Omron Connected devices. Each Omron device item in list has below properties.

KEY	DESCRIPTION
OMRONBLEConfigDevice.ModelName	Device Sales Name
OMRONBLEConfigDevice.ModelSeries	Device Series Type
OMRONBLEConfigDevice.Users	No of users available in memory
OMRONBLEConfigDevice.Category	Device Category
OMRONBLEConfigDevice.GroupID	Device Category Type
OMRONBLEConfigDevice.GroupIncludedGroupID	Device Model Type
OMRONBLEConfigDevice.Identifier	Device Identifier / Product Code
OMRONBLEConfigDevice.Protocol	Device Communication Protocol Standard
OMRONBLEConfigDevice.Image	Device main image path

OMRONBLEConfigDevice.Thumbnail	Device thumbnail image path

Device image and thumbnail image can be used by `OMRONBLEConfigDevice.Image` and `OMRONBLEConfigDevice.Thumbnail.` Asset aar is needed to be added first into the project before using these keys to show device images and thumbnails. See section V 3

```
List<HashMap<String, String>> deviceList = (List<HashMap<String,
String>>)OmronPeripheralManager.sharedManager(ctx).retrieveManagerConfigurati
on(context).get(OmronConstants.OMRONBLEConfigDeviceKey);
HashMap<String,String> item = deviceList.get(0);
if(item.get(OmronConstants.OMRONBLEConfigDevice.Thumbnail) != null) {
    Resources res = context.getResources();
    int resourceId =
res.getIdentifier(item.get(OmronConstants.OMRONBLEConfigDevice.Thumbnail),
"drawable", context.getPackageName());
}
```

Library posts notification to partner application when configuration is ready for use. Partner application is required to listen to these. The library posts different status codes, listed below:

KEY	DESCRIPTION
OmronConstants.OMRONConfigStatus.OMRONConfigurationFileSuccess	Configuration setup success
OmronConstants.OMRONConfigStatus.OMRONConfigurationFileError	Configuration setup failure
OmronConstantts.OMRONConfigStatus.OMRONConfigurationFileUpdateError	Configuration upgrade failure

```
LocalBroadcastManager.getInstance(this).registerReceiver(mMessageReceiver,
        new IntentFilter(OmronConstants.OMRONBLEConfigDeviceAvailabilityNotification));

private BroadcastReceiver mMessageReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        // Get extra data included in the Intent
        final int status = intent.getIntExtra(OmronConstants.OMRONConfigurationStatusKey, 0);

        if(status == OmronConstants.OMRONConfigurationStatus.OMRONConfigurationFileSuccess) {

            Log.d(TAG, "Config File Extract Success");

        }else if(status == OmronConstants.OMRONConfigurationStatus.OMRONConfigurationFileError) {

            Log.d(TAG, "Config File Extract Failure");

        }else if(status ==
OmronConstants.OMRONConfigurationStatus.OMRONConfigurationFileUpdateError) {

            Log.d(TAG, "Config File 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 (2)

(c) User Hash Id (**mandatory**) – This is for authenticating connection between Omron blood pressure monitor and application during Pair and Transfer. This is preferably the user's login email address, which will be used in application 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.
Disclaimer : Guest data from device will also get transferred by enabling this. Use only if required.

(f) 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 activity device: Height, weight and stride
For weight scale: Height, date of birth, gender and DCI

Personal settings (OMRONDevicePersonalSettingsKey) will have the following details

KEY	DESCRIPTION	UNIT	DEVICE TYPE
OMRONDevicePersonalSettings. UserHeightKey	Height of the user	cm	Activity and Body Composition
OMRONDevicePersonalSettings. UserWeightKey	Weight of the user	kg	Activity
OMRONDevicePersonalSettings. UserStridetKey	Stride of the user	cm	Activity
OMRONDevicePersonalSettings. UserDateOfBirthKey	Date of birth of user	yyyymmdd	Body Composition
OMRONDevicePersonalSettings. UserGenderKey	Gender of the user	0-Female 1-Male	Body Composition
OMRONDevicePersonalSettings. WeightKey	Weight settings	-	Body Composition

OMORNDevicePersonalSettings. BloodPressureKey	Blood Pressure settings	-	Blood Pressure
OMRONDevicePersonalSettings. TargetStepsKey	Target number of steps	Count	Activity
OMRONDevicePersonalSettings. TargetSleepKey	Target sleep time	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 gender, date of birth, 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: `OmronPersonalSettings.WeightDCINotAvailable`.
Blood Pressure:
`OmronPersonalSettings.BloodPressureDCINotAvailable`

Blood pressure settings:
`OMRONDevicePersonalSettingsBloodPressureKey` which will have the following details

KEY	DESCRIPTION	UNIT
OMRONDevicePersonalSettings.BloodPressureDCIKey	DCI Value for Blood pressure	Integer
OMRONDevicePersonalSettings.BloodPressureTruReadEnableKey	TruRead Enable Key	*Refer Below
OMRONDevicePersonalSettings.BloodPressureTruReadIntervalKey	TruRead Interval Key	*Refer Below

`OMRONDevicePersonalSettings.BloodPressureTruReadEnableKey`
and
`OMRONDevicePersonalSettings.BloodPressureTruReadIntervalKey`
can have the following possible values

KEY	VALUE
OMRONDevicePersonalSettings.BloodPressureTruReadEnableKey	OMRONDevicePersonalSettingsBloodPressureTruReadStatus.Off and OMRONDevicePersonalSettingsBloodPressureTruReadStatus.On
OMRONDevicePersonalSettings.BloodPressureTruReadIntervalKey	OMRONDevicePersonalSettingsBloodPressureTruReadInterval.Interval15 OMRONDevicePersonalSettingsBloodPressureTruReadInterval.Interval30 OMRONDevicePersonalSettingsBloodPressureTruReadInterval.Interval60 OMRONDevicePersonalSettingsBloodPressureTruReadInterval.Interval120

Weight Settings:
`OMRONDevicePersonalSettingsWeightKey` will have the following details.

KEY	DESCRIPTION	UNIT
OMRONDevicePersonalSettings.WeightDCIKey	Key to set DCI Value	Integer

Please see the below table for possible use cases

DCI VALUE	DCI APP	NEW DCI (DEVICE)	RESULT
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
OMRONDeviceTimeSettings.FormatKey	Key to specify the device time format. It can have either of two values	"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
OMRONDeviceDateSettings.FormatKey	Key to specify the device date format. It can have either of two values	"OMRONDeviceDateFormat.MonthDay" - MMDD "OMRONDeviceDateFormat.DayMonth" - DDMM

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

Device distance settings (`OMRONDeviceDistanceSettingsKey`) will have the following details

KEY	DESCRIPTION	VALUES
OMRONDeviceDistanceSettings.UnitKey	Key to specify the device distance format. It can have either of two values	OMRONDeviceDistanceUnitKilometer" - Kilometer "OMRONDeviceDistanceUnitMile" - Miles

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

Device sleep settings (`OMRONDeviceSleepSettingsKey`) will have the following details

KEY	DESCRIPTION	VALUES
OMRONDeviceSleepSettings.AutomaticKey	Auto sleep setting	OMRONDeviceSleepAutomatic.On / OMRONDeviceSleepAutomatic.Off
OMRONDeviceSleepSettings.AutomaticStartTimeKey	Start time	0~23
OMRONDeviceSleepSettings.AutomaticStopTimeKey	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
OMRONDeviceAlarmSettings.DaysKey	Alarm days
OMRONDeviceAlarmSettings.TimeKey	Alarm time
OMRONDeviceAlarmSettings.TypeKey	Alarm type

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

KEY	DESCRIPTION
OMRONDeviceAlarmSettings.HourKey	Hour (Always in 24 hour format)
OMRONDeviceAlarmSettings.MinuteKey	Minute

Alarm days (`OMRONDeviceAlarmSettings.DaysKey`) will have the following key-values

KEY	DESCRIPTION	VALUE
OMRONDeviceAlarmSettings. SundayKey	Set Alarm for Sunday	OMRONDeviceAlarmStatus.On - ON OMRONDeviceAlarmStatus.Off - OFF
OMRONDeviceAlarmSettings. MondayKey	Set Alarm for Monday	OMRONDeviceAlarmStatus.On - ON OMRONDeviceAlarmStatus.Off - OFF
OMRONDeviceAlarmSettings. TuesdayKey	Set Alarm for Tuesday	OMRONDeviceAlarmStatus.On - ON OMRONDeviceAlarmStatus.Off - OFF
OMRONDeviceAlarmSettings. WednesdayKey	Set Alarm for Wednesday	OMRONDeviceAlarmStatus.On - ON OMRONDeviceAlarmStatus.Off - OFF
OMRONDeviceAlarmSettings. ThursdayKey	Set Alarm for Thursday	OMRONDeviceAlarmStatus.On - ON OMRONDeviceAlarmStatus.Off - OFF
OMRONDeviceAlarmSettings. FridayKey	Set Alarm for Friday	OMRONDeviceAlarmStatus.On - ON OMRONDeviceAlarmStatus.Off - OFF
OMRONDeviceAlarmSettings. SaturdayKey	Set Alarm for Saturday	OMRONDeviceAlarmStatus.On - ON OMRONDeviceAlarmStatus.Off - OFF

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

KEY	DESCRIPTION
OMRONDeviceAlarmType.Normal	Normal alarm
OMRONDeviceAlarmType.Measure	Blood pressure measurement alarm
OMRONDeviceAlarmType.Medication	Medication alarm

VII. Device Weight Settings (Body Composition Devices only - optional)

Device Weight Settings `OMRONDeviceWeightSettingsKey` will have the following details

KEY	DESCRIPTION	VALUE
OMRONDeviceWeightSettingsUnitKey	Key to specify device weight unit	OMRONDeviceWeightUnit.Kg - Kilogram OMRONDeviceWeightUnit.Lbs - Lbs OMRONDeviceWeightUnit.St - Stones

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
OMRONDeviceWeightUnit.Kg - Kilogram	Cms (Centimeters)
OMRONDeviceWeightUnit.Lbs - Lbs	Ft - Inch (Feet - Inch)
OMRONDeviceWeightUnit.St - Stones	Ft - Inch (Feet - Inch)

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(App.getInstance().getApplicationContext()).setConfiguration
(config);
```

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

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

4. Details of Setting Configurations with code examples

Configure settings for connectivity library:

```
//Peripheral configuration
OmronPeripheralManagerConfig peripheralConfig =
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext
()).getConfiguration();
// Filter device to scan and connect (optional)
    List<HashMap<String, String>> filterDevices = new ArrayList<>();
    filterDevices.add(modelDetails);
    peripheralConfig.deviceFilters = filterDevices;
// Set Scan timeout interval (optional)
peripheralConfig.timeoutInterval = Constants.CONNECTION_TIMEOUT;
// Set User Hash Id (mandatory)
peripheralConfig.userHashId = "<email_address_of_user>"; // Set logged in
user email
// Set configuration for OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext
()).setConfiguration(peripheralConfig);
```

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 from `OmronPeriperhalManager`.

```
//Get device details from configuration file
List<HashMap<String, String>> deviceList = (List<HashMap<String,
String>>)OmronPeripheralManager.sharedManager(ctx).retrieveManagerConfigurat
ion(context).get(OmronConstants.OMRONBLEConfigDeviceKey);
//Get the details of particular device
HashMap<String,String> modelDetails = deviceList.get(<index of selected
device>);
```

Configuration for Blood Pressure Devices:

```
//Blood Pressure settings
HashMap<String, Object> bloodPressurePersonalSettings = new HashMap<>();
bloodPressurePersonalSettings.put(OmronConstants.OMRONDevicePersonalSettings.BloodPressureTruReadEnableKey,
OmronConstants.OMRONDevicePersonalSettingsBloodPressureTruReadStatus.On);
bloodPressurePersonalSettings.put(OmronConstants.OMRONDevicePersonalSettings.BloodPressureTruReadIntervalKey,
OmronConstants.OMRONDevicePersonalSettingsBloodPressureTruReadInterval.Interval30);
HashMap<String, Object> settings = new HashMap<>();
settings.put(OmronConstants.OMRONDevicePersonalSettings.BloodPressureKey,bloodPressurePersonalSettings);

HashMap<String, HashMap> personalSettings = new HashMap<>();
ArrayList<HashMap> deviceSettings = new ArrayList<>();
personalSettings.put(OmronConstants.OMRONDevicePersonalSettingsKey, settings);
deviceSettings.add(personalSettings);

peripheralConfig.deviceSettings = deviceSettings;
// Set configuration for OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).setConfiguration(peripheralConfig);
```

Configuration for Activity Devices:
a) Device and personal settings

```
HashMap<String, String> settingsModel = new HashMap<String, String>();
settingsModel.put(OmronConstants.OMRONDevicePersonalSettings.UserHeightKey, "<Height (cm)>");
settingsModel.put(OmronConstants.OMRONDevicePersonalSettings.UserWeightKey, "<Weight (kg)>");
settingsModel.put(OmronConstants.OMRONDevicePersonalSettings.UserStrideKey, "<Stride (cm)>");

HashMap<String, HashMap> userSettings = new HashMap<>();
userSettings.put(OmronConstants.OMRONDevicePersonalSettingsKey, settingsModel);

ArrayList<HashMap> deviceSettings = new ArrayList<>();
deviceSettings.add(userSettings);

// Set Device Settings
peripheralConfig.deviceSettings = deviceSettings;

// Set configuration for OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).setConfiguration(peripheralConfig);
```

b) Alarm settings, Time and Date Settings:

```
// Alarm Settings
// Alarm 1 Time
HashMap<String, Object> alarmTime1 = new HashMap<String, Object>();
alarmTime1.put(OmronConstants.OMRONDeviceAlarmSettings.HourKey, "15");
alarmTime1.put(OmronConstants.OMRONDeviceAlarmSettings.MinuteKey, "33");
// Alarm 1 Day (SUN-SAT)
HashMap<String, Object> alarmDays1 = new HashMap<String, Object>();
alarmDays1.put(OmronConstants.OMRONDeviceAlarmSettings.SundayKey,
OmronConstants.OMRONDeviceAlarmStatus.Off);
alarmDays1.put(OmronConstants.OMRONDeviceAlarmSettings.MondayKey,
OmronConstants.OMRONDeviceAlarmStatus.Off);
alarmDays1.put(OmronConstants.OMRONDeviceAlarmSettings.TuesdayKey,
OmronConstants.OMRONDeviceAlarmStatus.Off);
alarmDays1.put(OmronConstants.OMRONDeviceAlarmSettings.WednesdayKey,
OmronConstants.OMRONDeviceAlarmStatus.Off);
alarmDays1.put(OmronConstants.OMRONDeviceAlarmSettings.ThursdayKey,
OmronConstants.OMRONDeviceAlarmStatus.On);
alarmDays1.put(OmronConstants.OMRONDeviceAlarmSettings.FridayKey,
OmronConstants.OMRONDeviceAlarmStatus.Off);
alarmDays1.put(OmronConstants.OMRONDeviceAlarmSettings.SaturdayKey,
OmronConstants.OMRONDeviceAlarmStatus.Off);
HashMap<String, Object> alarm1 = new HashMap<>()
alarm1.put(OmronConstants.OMRONDeviceAlarmSettings.DaysKey, alarmDays1);
alarm1.put(OmronConstants.OMRONDeviceAlarmSettings.TimeKey, alarmTime1);
alarm1.put(OmronConstants.OMRONDeviceAlarmSettings.TypeKey,
OmronConstants.OMRONDeviceAlarmType.Measure);
// Add Alarm1, Alarm2, Alarm3 to List
ArrayList<HashMap> alarms = new ArrayList<>();
alarms.add(alarm1);
HashMap<String, Object> alarmSettings = new HashMap<>();
alarmSettings.put(OmronConstants.OMRONDeviceAlarmSettingsKey, alarms);
ArrayList<HashMap> deviceSettings = new ArrayList<>();
deviceSettings.add(alarmSettings);
// Date Format
HashMap<String, Object> dateFormatSettings = new HashMap<String, Object>();
dateFormatSettings.put(OmronConstants.OMRONDeviceDateSettings.FormatKey,
OmronConstants.OMRONDeviceDateFormat.DayMonth);
HashMap<String, HashMap> dateSettings = new HashMap<>();
dateSettings.put(OmronConstants.OMRONDeviceDateSettingsKey, dateFormatSettings);
// Time Format
HashMap<String, Object> timeFormatSettings = new HashMap<String, Object>();
timeFormatSettings.put(OmronConstants.OMRONDeviceTimeSettings.FormatKey,
OmronConstants.OMRONDeviceTimeFormat.Time24Hour);
HashMap<String, HashMap> timeSettings = new HashMap<>();
timeSettings.put(OmronConstants.OMRONDeviceTimeSettingsKey, timeFormatSettings);

ArrayList<HashMap> deviceSettings = new ArrayList<>();
deviceSettings.add(dateSettings);
deviceSettings.add(timeSettings);

// Set Device Settings
peripheralConfig.deviceSettings = deviceSettings;

// Set configuration for OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).setCo
nfiguration(peripheralConfig);
```

c) Distance Settings and Sleep Settings:

```
// Distance Unit Format
HashMap<String, Object> dateUnitSettings = new HashMap<String, Object>();
dateUnitSettings.put(OmronConstants.OMRONDeviceDistanceSettings.UnitKey,
OmronConstants.OMRONDeviceDistanceUnit.Kilometer);
HashMap<String, HashMap> distanceSettings = new HashMap<>();
distanceSettings.put(OmronConstants.OMRONDeviceDistanceSettingsKey, dateUnitSettings);


// Sleep Settings
HashMap<String, Object> sleepTimeSettings = new HashMap<String, Object>();
sleepTimeSettings.put(OmronConstants.OMRONDeviceSleepSettings.AutomaticKey,
OmronConstants.OMRONDeviceSleepAutomatic.On);
sleepTimeSettings.put(OmronConstants.OMRONDeviceSleepSettings.StartTimeKey, "19");
sleepTimeSettings.put(OmronConstants.OMRONDeviceSleepSettings.StopTimeKey, "20");
HashMap<String, HashMap> sleepSettings = new HashMap<>();
sleepSettings.put(OmronConstants.OMRONDeviceSleepSettingsKey, sleepTimeSettings);

ArrayList<HashMap> deviceSettings = new ArrayList<>();
deviceSettings.add(distanceSettings);
deviceSettings.add(sleepSettings);
// Set Device Settings
peripheralConfig.deviceSettings = deviceSettings;

// Set configuration for OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).setConfiguration(peripheralCo
nfig);
```

Configuration for Body Composition Devices:

a) Device and personal settings

```
HashMap<String, Object> settingsModel = new HashMap<>();
HashMap<String, HashMap> userSettings = new HashMap<>();
HashMap<String, Object> personalWeightSettings = new HashMap<>();
settingsModel.put(OmronConstants.OMRONDevicePersonalSettings.UserHeightKey, "<Height
(cm)>);
settingsModel.put(OmronConstants.OMRONDevicePersonalSettings.UserGenderKey,
OmronConstants.OMRONDevicePersonalSettingsUserGenderType.Female);
settingsModel.put(OmronConstants.OMRONDevicePersonalSettings.UserDateOfBirthKey,weigh
tBundle.getString(Constants.bundleKeys.KEY_BUNDLE_DOB,"19000101"));

//Weight settings
personalWeightSettings.put(OmronConstants.OMRONDevicePersonalSettings.WeightDCIKey,10
0);
settingsModel.put(OmronConstants.OMRONDevicePersonalSettings.WeightKey,personalWeight
Settings);
userSettings.put(OmronConstants.OMRONDevicePersonalSettingsKey, settingsModel);

ArrayList<HashMap> deviceSettings = new ArrayList<>();
deviceSettings.add(userSettings);

peripheralConfig.deviceSettings = deviceSettings;

// Set configuration for OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).setCo
nfiguration(peripheralConfig);
```

b) Weight unit Setting

```
// Weight Settings
// Add other weight common settings if any
HashMap<String, Object> weightCommonSettings = new HashMap<>();
weightCommonSettings.put(OmronConstants.OMRONDeviceWeightSettings.UnitKey,
OmronConstants.OMRONDeviceWeightUnit.Kg);
HashMap<String, Object> weightSettings = new HashMap<>();
weightSettings.put(OmronConstants.OMRONDeviceWeightSettingsKey,weightCommonSettings);
settingsModel.put(OmronConstants.OMRONDevicePersonalSettings.WeightKey,weightSettings
);
userSettings.put(OmronConstants.OMRONDevicePersonalSettingsKey, settingsModel);

ArrayList<HashMap> deviceSettings = new ArrayList<>();
deviceSettings.add(userSettings);

peripheralConfig.deviceSettings = deviceSettings;

// Set configuration for OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).setCo
nfiguration(peripheralConfig);
```

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(App.getInstance().getApplicationContext()).startManager();
```

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

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

```
// Start Scanning for Devices using OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext
()).startScanPeripherals(new OmronPeripheralManagerScanListener() {

    @Override
    public void onScanCompleted(final ArrayList<OmronPeripheral>
peripheralList, final ErrorInfo resultInfo) {

    }
});
```

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 for Devices using OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext
()).stopScanPeripherals(new OmronPeripheralManagerStopScanListener() {
    @Override
    public void onStopScanCompleted(final ErrorInfo resultInfo) {

    }
});
```

6. Connecting Omron Connected Devices

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

```
// Pair to Device using OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext(
)).connectPeripheral(omronPeripheral, new
OmronPeripheralManagerConnectListener() {

            @Override
            public void onConnectCompleted(final OmronPeripheral peripheral,
final ErrorInfo resultInfo) {

            }
});
```

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 i.e if it is available or not.

```
// Pair to Device using OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext(
)).connectPeripheral(omronPeripheral,true, new
OmronPeripheralManagerConnectListener() {

            @Override
            public void onConnectCompleted(final OmronPeripheral peripheral,
final ErrorInfo resultInfo) {

            }
});
```

On the completion of the above method, the following method requires to be used to send the user number for which App is pairing with the device.

```
// Pair to Device using OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).resumeConnectP
eripheral(omronPeripheral,mSelectedUser, new OmronPeripheralManagerConnectListener() {

        @Override
        public void onConnectCompleted(final OmronPeripheral peripheral, final ErrorInfo resultInfo) {

        }
});
```

or

```
// Pair to Device using OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).resumeConnectP
eripheral(omronPeripheral,selectedUsersList, new OmronPeripheralManagerConnectListener() {

        @Override
        public void onConnectCompleted(final OmronPeripheral peripheral, final ErrorInfo resultInfo) {

        }
});
```

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 `connectPeripheral(omronPeripheral,true,omronPeripheralManagerConnectListener):`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(App.getInstance().getApplicationContext()).endConnectPeri
pheral(new OmronPeripheralManagerConnectListener() {
        @Override public void onConnectCompleted(final OmronPeripheral peripheral,final ErrorInfo
resultInfo) {
        }
    });
```

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 previously paired Omron Connected Devices. The application has to pass the user number for which data need to be transferred from blood pressure monitor. Each Omron Blood Pressure monitor has defined number of user types. Some devices are single user device and others are two 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 could not 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 complete 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:

```
//Create peripheral object with localname and UUID
OmronPeripheral peripheral = new OmronPeripheral(localName, uuid);

// Data Transfer from Device using OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).startDataTransfer
FromPeripheral(peripheralLocal, selectedUser, true, new
OmronPeripheralManagerDataTransferListener() {

    @Override
    public void onDataTransferCompleted(final OmronPeripheral peripheral, final ErrorInfo resultInfo) {

    }

});
```

Or

```
//Create peripheral object with localname and UUID
OmronPeripheral peripheral = new OmronPeripheral(localName, uuid);

// Data Transfer from Device using OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).startDataTransfer
FromPeripheral(peripheralLocal, selectedUsersList, true, new
OmronPeripheralManagerDataTransferListener() {

   @Override
   public void onDataTransferCompleted(final OmronPeripheral peripheral, final ErrorInfo resultInfo) {

   }

});
```

End Data Transfer and clear unsent flags for data:

```
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()
).endDataTransferFromPeripheral(new
OmronPeripheralManagerDataTransferListener() {
    @Override
    public void onDataTransferCompleted(final OmronPeripheral peripheral,
final ErrorInfo errorInfo) {

    }
});
```

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 permissions 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.

```
OmronPeripheral peripheral = new
OmronPeripheral(OmronConstants.OMRONThermometerMC280B, "");

OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).startR
ecording(peripheral, new OmronPeripheralManagerRecordSignalListener() {
    @Override
    public void onSignalStrength(double signalLevel) {

    }
}, new OmronPeripheralManagerRecordListener() {
    @Override
    public void onRecord(OmronPeripheral peripheral, OmronErrorInfo errorInfo) {

    }
})
```

```
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).stopRe
cording(null);
```

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 Partner application uses `startDataTransferFromPeripheral`.

```
peripheral.getVitalDataWithUser(<selected user>, new OmronPeripheralListener() {
    @Override
    public void onGetDeviceInformationCompleted(HashMap<String, String> deviceInfo, ErrorInfo
errorInfo) {
    }
    @Override
    public void onGetVitalDataCompleted(HashMap<String, Object> vitalData, ErrorInfo errorInfo)
{
        // Blood Pressure Data
        final ArrayList<HashMap<String, Object>> bloodPressureItemList =
(ArrayList<HashMap<String, Object>>)
vitalData.get(OmronConstants.OMRONVitalDataBloodPressureKey);
        if (bloodPressureItemList != null) {

        }
        // Activity Data
        final ArrayList<HashMap<String, Object>> activityList = (ArrayList<HashMap<String,
Object>>) vitalData.get(OmronConstants.OMRONVitalDataActivityKey);
        if (activityList!= null) {

        }
        // Sleep Data
        ArrayList<HashMap<String, Object>> sleepingData = (ArrayList<HashMap<String, Object>>)
vitalData.get(OmronConstants.OMRONVitalDataSleepKey);
        if (sleepingData != null) {

        }
        // Records Data
        ArrayList<HashMap<String, Object>> recordData = (ArrayList<HashMap<String, Object>>)
vitalData.get(OmronConstants.OMRONVitalDataRecordKey);
        if (recordData != null) {

        }
        // Weight Data
        ArrayList<HashMap<String, Object>> weightData = (ArrayList<HashMap<String, Object>>)
vitalData.get(OmronConstants.OMRONVitalDataWeightKey);
        if (weightData != null) {

        }
        // Wheeze Data
        ArrayList<HashMap<String, Object>> wheezeData = (ArrayList<HashMap<String, Object>>)
vitalData.get(OmronConstants.OMRONVitalDataWheezeKey);
        if (weightData != null) {

        }
        // Pulse Oximeter Data
        ArrayList<HashMap<String, Object>> oxygenData = (ArrayList<HashMap<String, Object>>)
vitalData.get(OmronConstants.OMRONVitalDataPulseOximeterKey);
        if (weightData != null) {

        }
        // Temperature Data
        ArrayList<HashMap<String, Object>> oxygenData = (ArrayList<HashMap<String, Object>>)
vitalData.get(OmronConstants.OMRONVitalDataTemperatureKey);
        if (weightData != null) {

        }
    }
});
```

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 data	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
OMRONVitalData.SystolicKey	Systolic Blood Pressure	mmHg
OMRONVitalData.DiastolicKey	Diastolic Blood Pressure	mmHg
OMRONVitalData.PulseKey	Pulse	bpm (beats per minute)
OMRONVitalData.MovementFlagKey	Movement Error Symbol	-
OMRONVitalData.CuffFlagKey (for display)	Cuff Wrap Guide	0 – unfit 1 - fit
OMRONVitalData.ConsecutiveMeasurementKey	TruRead Index	-
OMRONVitalData.ArtifactDetectionKey	Number of detected artifact	-
OMRONVitalData.IrregularFlagKey	Irregular Heart Beat Symbol	-
OMRONVitalData.StartDateKey	Date Time Unix Timestamp (UTC)	-
OMRONVitalData.CuffWrapDetectionFlagKey (not for display)	Cuff Wrap Guide	0 – unfit 1 - fit
OMRONVitalData.IrregularHeartBeatCountKey	No of irregular heartbeat	-
OMRONVitalData.MeasurementModeKey	Blood pressure measurement mode	-
OMRONVitalData.DisplayedErrorCodeNightModeKey	Blood pressure measurement error code when using NightView device	-

This 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 `OmronConstants.OMRONVitalData.` 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` 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 `OmronConstants.OMRONVitalData.ConsecutiveMeasurementKey` 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 is displayed as 97
96.4 is displayed as 96

Blood Pressure Measurement mode (OMRONVitalData.MeasurementModeKey):

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 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.

Different type of Activity data available are listed below. Each of these data can be retrieved using the keys mentioned in `OmronConstants.OMRONVitalData`

ACTIVITY TYPE	DESCRIPTION	UNIT
OMRONActivityData.StepsPerDay	Steps for a day	count
OMRONActivityData.AerobicStepsPerDay	Aerobic steps for a day	count
OMRONActivityData.WalkingCaloriesPerDay	Calories burnt in a day	kcal
OMRONActivityData.DistancePerDay	Distance covered in a day	km

OMRONActivityData.NormalStepsPerDay	Normal steps in a day	count
OMRONActivityData.JoggingStepsPerDay	Jogging steps in a day	count
OMRONActivityData.FastStepsPerDay	Fast steps in a day	count

Each of activity types contains below data

KEY	DESCRIPTION
OMRONActivityData.StartDateKey	Start Date Time Unix Timestamp (UTC)
OMRONActivityData.EndDateKey	End Date Time Unix Timestamp (UTC)
OMRONActivityData.MeasurementKey	Measurement
OMRONActivityData.SequenceKey	Sequence
OMRONActivityData.DividedDataKey	Hourly data (list)

Each of divided data contains below data

KEY	DESCRIPTION
OMRONActivityData.DividedDataStartDateKey	Start Date Time Unix Timestamp for Hourly data (UTC)
OMRONActivityData.DividedDataMeasurementKey	Measurements in Hourly data
OMRONActivityData.DividedDataPeriodTimeKey	Period in Hourly data
OMRONActivityData.DividedDataMeasurementDetailsKey	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. OMRONActivityData.SequenceKey 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. Each of these data can be retrieved using the keys mentioned in `OmronConstants.OMRONVitalData`

Available data present in sleep are listed below

KEY	DESCRIPTION
OMRONSleepData.StartDateKey	Start Date Time Unix Timestamp (UTC)
OMRONSleepData.EndDateKey	End Date Time Unix Timestamp (UTC)
OMRONSleepData.TimeInBedKey	Time in bed – Unix Timestamp (UTC)
OMRONSleepData.SleepOnsetTimeKey	Sleep Time – Unix Timestamp (UTC)
OMRONSleepData.WakeTimeKey	Wakeup Time – Unix Timestamp (UTC)
OMRONSleepData.TotalSleepTimeKey	Total Sleep Time in minutes
OMRONSleepData.SleepEfficiencyKey	Efficiency of sleep in percentage
OMRONSleepData.ArousalDuringSleepTimeKey	Minutes awake
OMRONSleepData.BodyMotionLevelKey	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. Use key OMRONVitalDataRecordKey to get records from Vital Data list

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 dictionary of different weight types like weight value, BMI, Body age, Visceral fat percentage and so on. Each of these data can be retrieved using the keys mentioned in `OmronConstants.OMRONVitalData`

Available data present in weight are listed below

KEY	DESCRIPTION	UNIT/ RANGE
OMRONWeightData.StartDateKey	Start Date Time Unix Timestamp	UTC
OMRONWeightData.SequenceKey	Sequence Number	Integer
OMRONWeightData.UserIdKey	User id	1-4
OMRONWeightData.WeightKey	Weight	2-150 kg
OMRONWeightData.BodyFatLevelClassificationKey	Body fat classification	-
OMRONWeightData.BodyFatPercentageKey	Body fat percentage	5.0-60
OMRONWeightData.RestingMetabolismKey	Resting metabolism	385-3999
OMRONWeightData.SkeletalMusclePercentageKey	Skeletal muscle percentage	5.0-50
OMRONWeightData.BMIKey	Body mass index	7-90
OMRONWeightData.VisceralFatLevelKey	Visceral fat level	Upto 30
OMRONWeightData.VisceralFatLevelPercentageKey	Visceral fat percentage	Upto 4 Levels
OMRONWeightData.SkeletalMuscleLevelClassification Key	SkeletalMusclelevel 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 `OmronConstants.OMRONVitalData`

KEY	DESCRIPTION	Range/Unit
OMRONWheezeData.StartDateKey	Measurement Start Date	timeStamp
OMRONWheezeData.SequenceKey	Sequence Number	Int
OMRONWheezeData.UserIdKey	User id	1
OMRONWheezeData.WheezeKey	Wheeze data	0 : Detected 1 : Not detected 2 : Measurement Error
OMRONWheezeData.ErrorNoiseKey	Excessive Noise Error	0 : No error 1 : Error
OMRONWheezeData.ErrorDecreaseBreathingSound LevelKey	Decrease in breathing sound error	0 : No error 1 : Error
OMRONWheezeData.ErrorSurroundingNoiseKey	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 `OmronConstants.OMRONVitalData`

KEY	DESCRIPTION	Range/Unit
OMRONPulseOximeterData.StartDateKey	Measurement Start Date	timeStamp
OMRONPulseOximeterData.SequenceKey	Sequence Number	Int
OMRONPulseOximeterData.UserIdKey	User id	1
OMRONPulseOximeterData.SPO2LevelKey	Blood oxygen level	Percentage
OMRONPulseOximeterData.PulseRateKey	Pulse rate	bpm
OMRONPulseOximeterData.AmplitudeKey	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 `OmronConstants.OMRONVitalData`

KEY	DESCRIPTION	Range/Unit
OMRONTemperatureData.StartDateKey	Measurement Start Date	timeStamp
OMRONTemperatureData.SequenceKey	Sequence Number	Int

OMRONTemperatureData.UserIdKey	User id	1
OMRONTemperatureData.TemperatureKey	Temperature value (LCD display)	-
OMRONTemperatureData.TemperatureCelsiusKey	Temperature value in Celsius	bpm
OMRONTemperatureData.TemperatureUnitKey	Unit of Temperature	-Celsius / Fahrenheit
OMRONTemperatureData.TemperatureLevelKey	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.

```
// Disconnect device using OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext
()).disconnectPeripheral(mSelectedPeripheral, new
OmronPeripheralManagerDisconnectListener() {
    @Override
    public void onDisconnectCompleted(OmronPeripheral peripheral, ErrorInfo
resultInfo) {

    }
});
```

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) and add a maximum of 5 alarms. 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(App.getInstance().getApplicationContext
()).getConfiguration();
// Add all required settings for the peripheral
peripheralConfig.deviceSettings = deviceSettings;

// Set configuration for OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext
()).setConfiguration(peripheralConfig);

//Call to update the settings
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext
()).updatePeripheral(peripheral, new OmronPeripheralManagerUpdateListener()
{
    @Override
    public void onUpdateCompleted(final OmronPeripheral peripheral, final
ErrorInfo resultInfo) {

    }
});
```

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(App.getInstance().getApplicationContex
t()).getConfiguration();
// Add all required settings for the peripheral
peripheralConfig.deviceSettings = deviceSettings;

// Set configuration for OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContex
t()).setConfiguration(peripheralConfig);//Call to update the settings with
selected user <user 1>
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContex
t()).updatePeripheral(peripheral, selectedUser, new
OmronPeripheralManagerUpdateListener() {
    @Override
    public void onUpdateCompleted(final OmronPeripheral peripheral, final
ErrorInfo resultInfo) {

    }
});
//or start updating device with users list
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContex
t()).updatePeripheral(peripheral, selectedUserList , new
OmronPeripheralManagerUpdateListener() {
    @Override
    public void onUpdateCompleted(final OmronPeripheral peripheral, final
ErrorInfo resultInfo) {

    }
});
```

Please refer table under Appendix III 3 (d) 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.

```
peripheral.getDeviceInformation(
        new OmronPeripheralListener.DeviceInformation() {
            @Override
            public void onGetDeviceInformationCompleted(
                    HashMap<String, String> deviceInfo,
                    ErrorInfo errorInfo) {

            }
        });
```

Alternatively

```
HashMap<String, String> deviceInformation = peripheral.getDeviceInformation();
```

This information can be retrieved using keys defined in `OmronConstants.OMRONDeviceInformation`.

KEY	DESCRIPTION	DEVICE TYPE
OMRONDeviceInformationDisplayNameKey	Display Name	All
OMRONDeviceInformationIdentitiyNameKey	Identity name	All
OMRONDeviceInformationLocalNameKey	Local Name	All
OMRONDeviceInformationSerialIdKey	Device Serial Id	All
OMRONDeviceInformationUUIDKey	Device UUID	All

b) Device Settings

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

```
ArrayList<HashMap>  deviceSettings  = peripheral.getDeviceSettings();
```

The above method provide the below details

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

Data received from above method will include all information related to device settings, available users, registered users and users personal settings. For sample response, refer Appendix:VIII 3 (c). User's personal settings are under key `OMRONDevicePersonalSettingsKey`

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

These personal settings also have the information about display settings i.e enable/disable various items in display and the priority of display settings.
The enable/disable keys provide information about the different items of device, if the display of those are enabled or disabled and the priority order tells about the order in which these items are displayed on the device.

Following table contains the required information about these keys

KEY	DESCRIPTION
OMRONDevicePersonalSettings.WeightDisplayPriorityBodyFatKey	Display priority – Body fat
OMRONDevicePersonalSettings.WeightDisplayPriorityVisceralFatLevelKey	Display priority – Visceral fat
OMRONDevicePersonalSettings.WeightDisplayPrioritySkeletalMuscleLevelKey	Display priority – Skeletal muscle
OMRONDevicePersonalSettings.WeightDisplayPriorityRestingMetabolismKey	Display priority – Resting metabolism
OMRONDevicePersonalSettings.WeightDisplayPriorityBMIKey	Display priority – BMI
OMRONDevicePersonalSettings.WeightDisplayEnableBodyFatKey	Display enable – Body fat
OMRONDevicePersonalSettings.WeightDisplayEnableVisceralFatLevelKey	Display enable – Visceral fat
OMRONDevicePersonalSettings.WeightDisplayEnableSkeletalMuscleLevelKey	Display enable – Skeletal muscle
OMRONDevicePersonalSettings.WeightDisplayEnableRestingMetabolismKey	Display enable – Resting metabolism
OMRONDevicePersonalSettings.WeightDisplayEnableBMIKey	Display enable - BMI

To get settings data for a particular user, following method can be used that requires a user number as well. This will return the personal settings of the particular user.

```
Object personalSettingsForUser1 = peripheral.getDeviceSettingsWithUser(<selected user>);
```

Alternatively, following method can be used with a call back method that will provide device settings and an error info if something went wrong.

```
peripheral.getDeviceSettingsWithUser(
         <selected user>,
         new OmronPeripheralListener.DeviceSettings() {
             @Override
             public void onGetDeviceSettingsCompleted(
                     Object deviceSettings,
                     ErrorInfo errorInfo) {

             }
         });
```

c) Bluetooth state change Information

Application can listen to Bluetooth events when communicating with Omron Connected Devices. Two additional features provided by the 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
LocalBroadcastManager.getInstance(this).registerReceiver(mMessageReceiver,newIntentFilter(O
mronConstants.OMRONBLECentralManagerDidUpdateStateNotification));
private BroadcastReceiver mMessageReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {

        // Get extra data included in the Intent
        int status = intent.getIntExtra(OmronConstants.OMRONBLEBluetoothStateKey, 0);

        if (status ==
OmronConstants.OMRONBLEBluetoothState.OMRONBLEBluetoothStateUnknown) {

            Log.d(TAG, "Bluetooth is in unknown state");

        } else if (status ==
OmronConstants.OMRONBLEBluetoothState.OMRONBLEBluetoothStateOff) {

            Log.d(TAG, "Bluetooth is currently powered off");

        } else if (status ==
OmronConstants.OMRONBLEBluetoothState.OMRONBLEBluetoothStateOn) {

            Log.d(TAG, "Bluetooth is currently powered on");
        }
    }
};
```

II. Bluetooth device state changes

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

```
// Listen to Device state changes using OmronPeripheralManager
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationCon
text()).onConnectStateChange(new
OmronPeripheralManagerConnectStateListener() {

    @Override
    public void onConnectStateChange(final int state) {

        runOnUiThread(new Runnable() {
            @Override
            public void run() {

                String status = "-";

                if (state ==
OmronConstants.OMRONBLEConnectionState.CONNECTING) {
                    status = "Connecting...";
                } else if (state ==
OmronConstants.OMRONBLEConnectionState.CONNECTED) {
                    status = "Connected";
                } else if (state ==
OmronConstants.OMRONBLEConnectionState.DISCONNECTING) {
                    status = "Disconnecting...";
                } else if (state ==
OmronConstants.OMRONBLEConnectionState.DISCONNECTED) {
                    status = "Disconnected";
                }
            }
        });
    }
});
```

d) Get library version

Partner application can check the current version of the library using below.

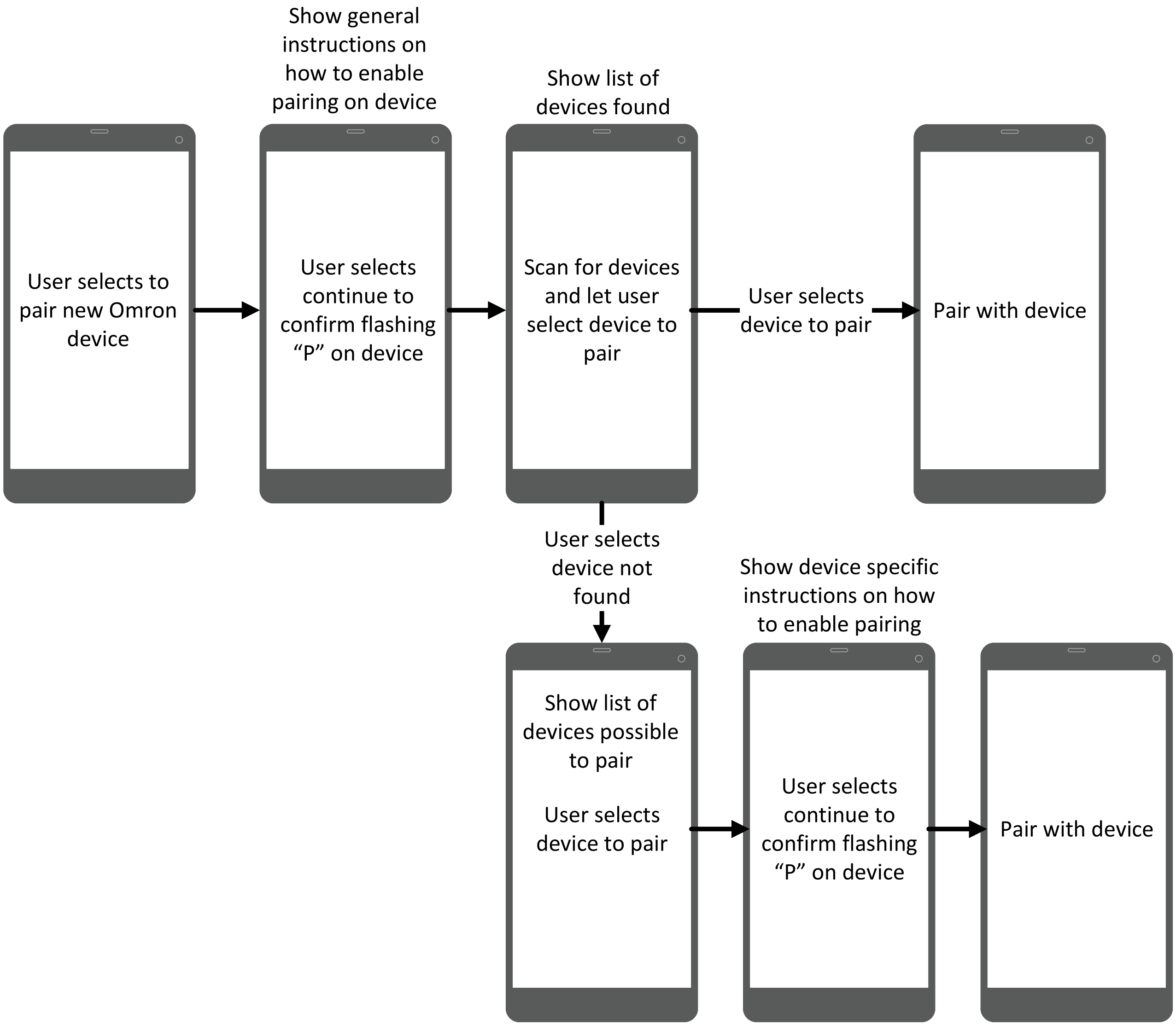
```
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext()).libVersion()
```

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.


Show general instructions on how to enable pairing on device
Show list of devices found
User selects to pair new Omron device
User selects continue to confirm flashing “P” on device
Scan for devices and let user select device to pair
User selects device to pair
Pair with device
User selects device not found
Show device specific instructions on how to enable pairing
Show list of devices possible to pair
User selects device to pair
User selects continue to confirm flashing “P” on device
Pair with device

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

Step 2 in “OmronConnectivityLibrary Library 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 properties, which will help identify details of device.

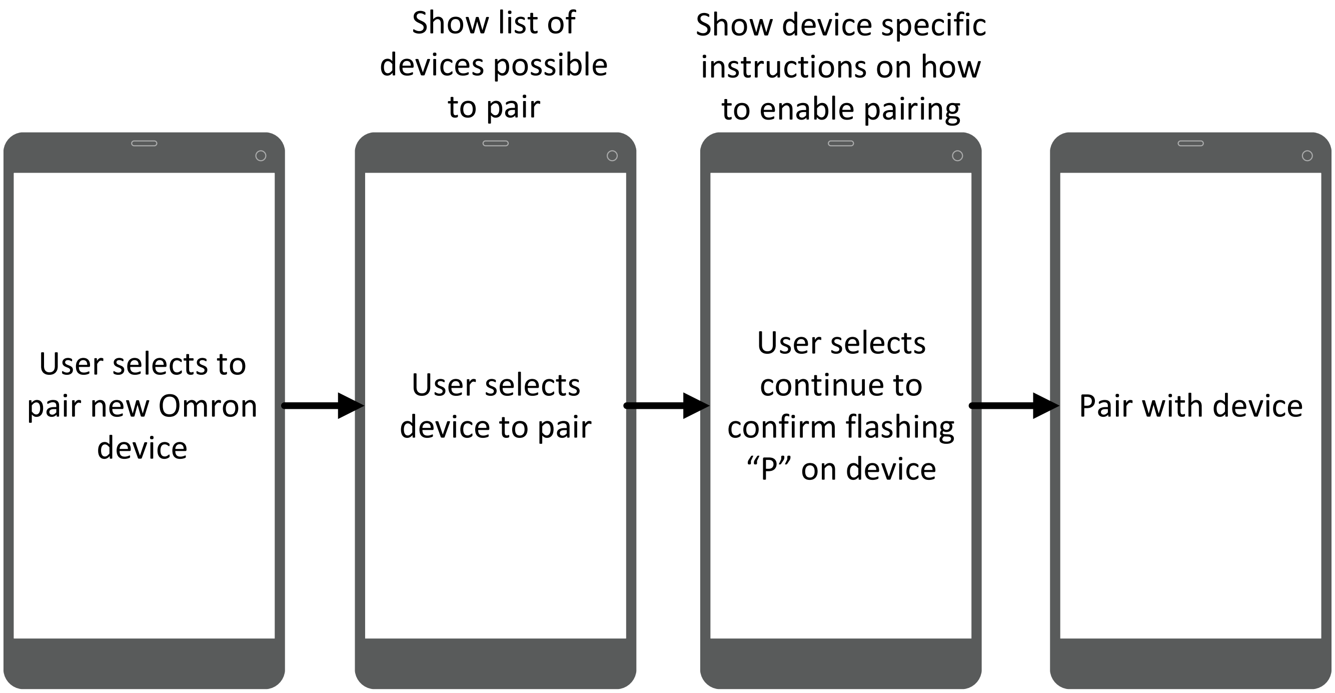
KEY
OMRONBLEConfigDevice.GroupID
OMRONBLEConfigDevice.GroupIncludedGroupID

Below function can be used to retrieve device configuration.

```
OmronPeripheralManagerConfig peripheralConfig = 
OmronPeripheralManager.sharedManager(App.getInstance().getApplicationContext())
.getConfiguration();
Log.d(TAG, "Device Config :  " + 
peripheralConfig.getDeviceConfigGroupIdAndGroupIncludedId(peripheral.getDeviceG
roupIDKey(), peripheral.getDeviceGroupIncludedGroupIDKey()));
```

2. Pairing: Scan for only selected Omron Connected Devices

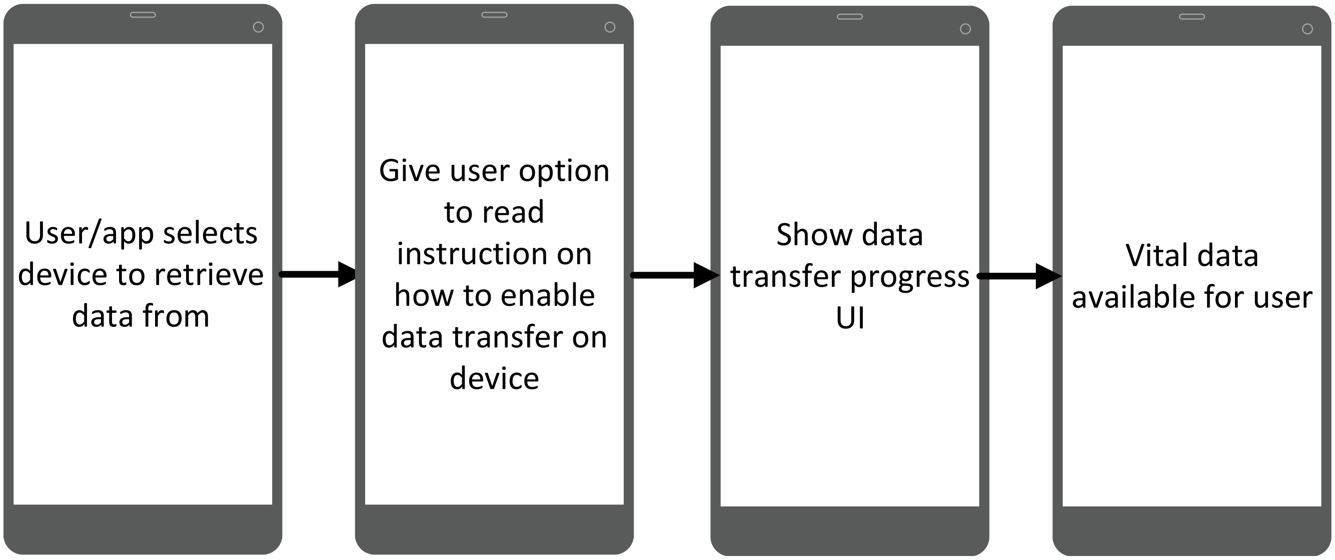
Partner application can choose to scan only for selected Omron Connected Device model. This will allow OmronPeripheralManager to filter discovering Omron Connected Devices.


Show list of devices possible to pair
Show device specific instructions on how to enable pairing
User selects to pair new Omron device
User selects device to pair
User selects continue to confirm flashing "P" on device
Pair with device

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.


User/app selects device to retrieve data from
Give user option to read instruction on how to enable data transfer on device
Show data transfer progress UI
Vital data available for user

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 (ErrorInfo). 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. 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 library.

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	Bluetooth Off
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;
                };
};
```