Shine SDK for Android

Version 2.3

Table of Contents

Overview

Shine SDK for Android provides a set of APIs that help you build native applications that communicate directly with Misfit Shine and Misfit Flash.

Key Terms and Concepts

Serial Number

A unique identifier associates with a Shine/Flash device. The serial number is set only once during the manufacturing process.

Activity

Representing the user’s movement in one minute, which includes the steps count and activity points.

Point

A metric used to measure the amount of movement of physical activities. It is calculated based on the step count and step frequency, which enables the accurate measurement of the energy spent on doing the activities. The Shine/Flash device will give you a precise assessment (in terms of Activity Points) of how much you moved, and this measurement is very consistent from day to day so that you can have confidence in comparing between days and towards your goal.

Goal

The daily target that user has set to achieve. It is measured in points.

Firmware

Shine/Flash's firmware. The firmware can be updated via OTA (Over-The-Air update).

Basic Operations

Scan

Search for all Shine and Flash nearby. The application will receive a callback for each of the Shine/Flash that is found. Each one is associated with a unique serial number.

Connect

Establishing a connection to Shine/Flash via ***ShineDevice.connectProfile()***. The returned ***ShineProfile*** object can be used to further interact with the device.

Configure

The application can configure the clock, progress and daily goal on Shine/Flash simply by calling **ShineProfile.startSettingDeviceConfiguration().** The clock and timezone on the device will also be calibrated automatically in the process.

Sync

Read minute-by-minute movement data from the device. The data that was read will be deleted automatically afterwards.

OTA (Over-The-Air update)

Update device's firmware. New firmwares will be released regularly via Misfit OpenAPI or other channels.

Activate Flash

New Flash from the factory are put in IDLE mode to save battery. These units need to be activated to sync with other devices.

There're two ways to activate the Flash.

- Pressing the button on Flash. This will temporary put the device in ACTIVATED mode for a few hours.
- Programmatically activate the device via an API provided in ShineProfile class. This will permanently put the device in ACTIVATED mode.

Manual activation should be is required on the first sync only. The mobile application is recommended to permanently activate the device so that users no longer have to do it manually on subsequence syncs. This will also unlock the ability to do automatic and background syncs.

Supported Devices

Shine SDK supports any Android devices that has Bluetooth 4.0 and run Android OS 4.3 and later. Samsung devices running Android OS 4.2.2 and above are also supported.

API Documentation

ShineAdapter

Class Overview

ShineAdapter provides basic APIs to scan for Shine and Flash nearby.

Public Method

Return Type	Method

ShineAdapter	**getDefaultAdapter(Context context)** *Get the shared instance of ShineAdapter.*
boolean	**isEnabled()** *Check the bluetooth device is enabled or not.*
void	**startScanning(ShineScanCallback callback)** *Start scanning for Shine and Flash nearby.* ▪ *@param callback: to get notified when a Shine/Flash is found.*
void	**startScanning(ScanSettings settings, ShineScanCallback callback)** *Start scanning with power optimization for Shine and Flash nearby.* ▪ *@param settings: to modified power when scanning* ▪ *@param callback: to get notified when a Shine/Flash is found.*
void	**stopScanning(ShineScanCallback callback) throws IllegalArgumentException** *Stop an ongoing scan.* ▪ *@param callback: the callback instance that was passed to startScanning.*
void	**getConnectedShines(ShineRetrieveCallback callback) throws IllegalArgumentException** *Get list of Shines/Flashes currently connected to the phone, including those connected by other applications.* ▪ *@param callback: to receive the list of connected devices.* ▪ *@throws IllegalArgumentException: if callback is null.*
void	**getHIDConnectedShines(ShineRetrieveCallback callback) throws IllegalArgumentException** *Get list of Shines/Flashes currently connected to the phone with HID Over GATT, including those connected by other applications.* ▪ *@param callback: to receive the list of HID connected devices.* ▪ *@throws IllegalArgumentException: if callback is null.*

ShineScanCallBack

Class Overview

ShineScanCallback is used to receive the scan result of *ShineAdapter*.

Public Method

Return Type	Method
void	**onScanResult(ShineDevice device, int rssi)** *The callback will be invoked for each of the Shine/Flash found. There might be multiple callbacks for the same device.* ▪ *@param device: the Shine/Flash that is found.* ▪ *@param rssi: signal strength.*
void	**onScanFailed(ShineAdapter.ScanFailedErrorCode errorCode)** *The callback will be invoked if scanning failed.* ▪ *@param errorCode: error code of scanning failed.*

ShineAdapter.ScanFailedErrorCode

Class Overview

Enum of error code when onScanFailed callback is invoked

Constant	Value	Meaning
ALREADY_STARTED	1	Fails to start scan as BLE scan with the same settings is already started by the app
REGISTRATION_FAILED	2	Fails to start scan as app cannot be registered.
INTERNAL_ERROR	3	Fails to start scan due an internal error
FEATURE_UNSUPPORTED	4	Fails to start power optimized scan as this feature is not supported.
SDK_VALIDATION_FAILED	5	Fails to start scan because of SDK validation failed

ShineRetrieveCallback

Class Overview

ShineRetrieveCallback is used to retrieve number of connected shines of ShineAdapter

Public Method

Return Type	Method
void	**onConnectedShinesRetrieved(List<ShineDevice> connectedShines)** *The callback will be invoked when get connected shines* • *@param connectedShines: list of connected shines*

ShineDevice

Class Overview

ShineDevice represents the physical Shine/Flash device.

Public Method

Type	Method
ShineProfile	**connectProfile(Context context, boolean autoConnect, ShineProfile.ConnectionCallback connectionCallback)** *re-connect to remote Shine/Flash. This method is used to re-establish the connection to a disconnected device* • *@param connectionCallback: refer to* **ShineProfile.ConnectionCallback** • *@return: indicates the success or failure of this operation.*
String	**getSerialNumber()** *Return the unique identifier of the device.* • *@return: the device's serial number.*
String	**getName()** *Return bluetooth name.* • *@return: the device's bluetooth name.*
String	**getAddress()** *Return bluetooth mac address.* • *@return: the device's bluetooth mac address.*

String	**getBondState()** *Return bonding state.* • *@return: the device's bonding state, which can be either BluetoothDevice.BOND_NONE, BluetoothDevice.BOND_BONDING or BluetoothDevice.BOND_BONDED.*
ShineProfile	**getShineProfile()** *Get an existing ShineProfile of this device.* • *@return: the ShineProfile object of this device. Null if the profile is not created yet or had already been closed.*

ShineProfile

Class Overview

ShineProfile provides all the APIs needed to configure, read data and update the firmware on Shine/Flash.

Constant

Profile State Constant Enumeration

Type	Constant	Value	Description
ShineProfile.State	IDLE	0	Idle
ShineProfile.State	CONNECTING	1	Connecting Shine/Flash
ShineProfile.State	CONNECTED	2	Shine/Flash is connected.
ShineProfile.State	OTA	3	Update firmware Shine/Flash via OTA
ShineProfile.State	DISCONNECTING	4	Disconnecting Shine/Flash
ShineProfile.State	CLOSED	5	The connection is closed. This instance of ShineProfile is no longer usable.

Device Family Code

Constant	Description
DEVICE_FAMILY_UNKNOWN	Unknown device OR this information is not available yet.
DEVICE_FAMILY_SHINE	Shine.
DEVICE_FAMILY_FLASH	Flash.
DEVICE_FAMILY_BUTTON	Button.

Public Method

Type	Method
State	**getState()** • *@return: the profile state (refer to **ShineProfile.State**)*
String	**getFirmwareVersion()** • *@return: the current firmware version. This information is only available when the device is connected.*
String	**getModelNumber()** • *@return: model number of the device. This information is only available when the device is connected.*
String	**getDeviceFamily()** • *@return: device family (refer to **Device Family Code**). This information is only available when the device is connected.*

boolean	**playAnimation(ShineProfile.ConfigurationCallback configurationCallback)** *Play the LED animation on Shine/Flash.* • *@param configurationCallback refer to* ***ShineProfile.ConfigurationCallback*** • *@return: indicates the success or failure of this operation.*
boolean	**stopPlayingAnimation(ShineProfile.ConfigurationCallback configurationCallback)** *Stop playing the LED animation on Shine/Flash* • *@param: configurationCallback refer to* ***ShineProfile.ConfigurationCallback*** • *@return: indicates the success or failure of this operation.*
boolean	**sync(ShineProfile.SyncCallback syncCallback)** *Start a new sync session with Shine/Flash.* • *@param: syncCallback refer to* ***ShineProfile.SyncCallback*** • *@return: indicates the success or failure of this operation.*
boolean	**ota(byte[] firmwareData, ShineProfile.OTACallback otaCallback)** *Flash new firmware to Shine/Flash.* • *@param firmwareData: binary data of the firmware.* • *@param: otaCallback refer to* ***ShineProfile.OTACallback*** • *@return: indicates the success or failure of this operation.*
boolean	**getDeviceConfiguration(ShineProfile.ConfigurationCallback configurationCallback)** *Start getting the configuration. The current configuration will be returned via the callback.* • *@param: configurationCallback refer to* ***ShineProfile.ConfigurationCallback*** • *@return: indicates the success or failure of this operation.*
boolean	**setDeviceConfiguration(ConfigurationSession configurationSession, ShineProfile.ConfigurationCallback configurationCallback)** *Start updating the configuration which includes clock state, timestamp, timezone, daily goal, progress and other configurable properties.* • *@param: configurationSession refer to* ***ConfigurationSession*** • *@param: configurationCallback refer to* ***ShineProfile.ConfigurationCallback*** • *@return: indicates the success or failure of this operation.*
boolean	**activate(*ShineProfile.*ConfigurationCallback configurationCallback)** *Activate Flash, permanently put it in ACTIVATED mode.* • *@param: configurationCallback refer to* ***ShineProfile.ConfigurationCallback*** • *@return: indicates the success or failure of this operation.*
boolean	**getActivationState(*ShineProfile.*ConfigurationCallback configurationCallback)** *Start getting Flash's activation state. The current state will be returned via the callback.* • *@param: configurationCallback refer to* ***ShineProfile.ConfigurationCallback*** • *@return: indicates the success or failure of this operation.*
boolean	**readRssi(*ShineProfile.*ConfigurationCallback configurationCallback)** *Read the RSSI of the current device, and the result will be returned in the onRssiReadSucceeded(int) callback function of ShineProfileCallBack* • *@param: configurationCallback refer to* ***ShineProfile.ConfigurationCallback*** • *@return: indicates the success or failure of this operation.*
boolean	**interrupt()** *Interrupt current action* • *@return: indicates the success or failure of this operation.*

ShineDevice	**getDevice()** • *@return: remote Shine/Flash associated with this instance of ShineProfile.*
ActionID	**getCurrentAction()** *Get current action* • *@return: action ID of this operation.*
void	**close()** *Must be called to cleanup the ShineProfile instance and release all the occupied resources. After being closed, this instance of ShineProfile is no longer usable.* *In order to connect to the device again, the application must create a new instance via ShineDevice.connectProfile().*

ShineProfile.ConnectionCallback

Class Overview

The callback to receive the result of state changing in ShineProfile.

Type	Method
void	**onConnectionStateChanged(ShineProfile shineProfile, ShineProfile.State newState)** *Will be invoke when application's state change* • *@param: shineProfile, refer to ShineProfile* • @param: new State for application, refer to ShineProfile.State

ShineProfile.ConfigurationCallback

Class Overview

The callback to receive the result of configuration operations in ShineProfile.

Type	Method
void	**onConfigCompleted(ActionID actionID, ShineCallBackResult resultCode, Hashtable<ShineProperty, Object> data)** *Will be invoked after the configuration command has completed successfully.* • *@param actionID: id of every configuration command, refer to **ActionID*** • *@param resultCode: refer to **ShineCallBackResult*** • *@param data: data of application returning*

ShineProfile.SyncCallback

Class Overview

The callback to receive the result of sync operation in ShineProfile.

Type	Method

void	**onSyncDataRead(SyncResult syncResult, Bundle extraInfo, MutableBoolean shouldStop)** • *@param syncResult: contains an array of minute-by-minute activities.* • *@param extraInfo: extra information* • *@param shouldStop: whether to interrupt the sync. "true": stop the sync and keep this batch of activities on the device; "false": erase this batch from the device and continue. It is set to "false" by default.*
void	**onSyncCompleted(ShineCallBackResult resultCode)** *Will be invoke when sync completed* • *@param resultCode: refer to* **ShineCallBackResult**

ShineProfile.OTACallback

Class Overview

The callback to receive the result of OTA update operation in ShineProfile.

Type	Method
void	**onOTACompleted(ShineCallBackResult resultCode)** *Will be invoked after successfully completed the OTA update. The connection will be closed automatically right after this event.* • *@param resultCode: refer to* ***ShineCallBackResult***
void	**onOTAProgressChanged(float progress)** *Will be invoked to inform the application of the OTA progress.* • *@param progress: the progress of OTA process. Its value ranges from 0.0 to 1.0.*

ActionResult

Class Overview

Enum of results of callbacks are invoked after every action

Constant	Meaning
SUCCEEDED	Action succeed
FAILED	Action failed
TIMED_OUT	Action timed out
INTERNAL_ERROR	Action internal error
INTERRUPTED	Action interrupted
UNSUPPORTED	Action unsupported by firmware

ActionID

Class Overview

Enumeration for every action identification

Constant	Meaning
ANIMATE	Play Animation
ACTIVATE	Activate Flash

GET_ACTIVATION_STATE	Get Activation State
GET_CONFIGURATION	Get Configuration
SET_CONFIGURATION	Set Configuration
SYNC	Sync
OTA	Update Firmware via OTA
GET_CONNECTION_PARAMETERS	Get Connection Parameters
SET_CONNECTION_PARAMETERS	Set Connection Parameters
READ_REMOTE_RSSI	Read Remote RSSI

ShineProperty

Class Overview

Enumeration for getting data from the HashTable from onConfigCompleted

Constant	Meaning
SHINE_CONFIGURATION_SESSION	Get shineConfiguration value, refer to **ConfigurationSession**
ACTIVATION_STATE	Get activationState value, refer to **boolean**
RSSI	Get rssi value, refer to **int**

Activity

Class Overview

This class representing the user’s movement in one minute, which includes the steps count and activity points.

Public Variable

Type	Variable
long	mStartTimestamp *The start timestamp of this activity.*
long	mEndTimestamp *The end timestamp of this activity.*
int	mPoints *The total activity points user earned during this activity.*
int	mBipedalCount *The total bipedal count during this activity.*

ShineConfiguration

Class Overview

ShineConfiguration represents the configuration of the device, which consist of the Activity Points, Goal, Clock State and Battery Level.

Public Variable

Type	**Variable**
int	mActivityPoint *The current activity points.*
int	mGoalValue *The daily target points.*
int	mClockState *The Clock state, there are 3 states. Clock Enabled, Clock Disabled, Clock shown before progress.*
int	mBatteryLevel *Percentage of remaining battery capacity, value range 0-100.*

Profile State Constant

Type	**Constant**	**Description**
int	CLOCK_STATE_DISABLE	When user double tap the Shine/Flash, shows the daily progress only.
int	CLOCK_STATE_ENABLE	When user double tap the Shine/Flash, shows the daily progress then the clock.
int	CLOCK_STATE_SHOW_CLOCK_FIRST	When user double tap the Shine/Flash, shows the clock before daily progress.

SyncResult

Class Overview

SyncResult stores result of the sync operation.

Type	**Method**
ArrayList<Activity>	mActivities *An array of minute-by-minute activities.*

SDKSetting

Class Overview

SDKSetting stores necessary settings for Shine Android SDK, including setting user ID to identify 3rd party app’s user.

Type	**Method**
void	**setUp(Context applicationContext, String userId) throws IllegalArgumentException** *Must be called before using other APIs in the SDK.* ▪ *@param userId: an unique string associated with current user in your system. It may not be longer than 30 characters and only consist of alphanumeric characters (A-Z, a-z, 0-9) and some special characters such as dot (.), dash (-), underscore(_) and at (@).* ▪ *@param applicationContext.* ▪ @throws *IllegalArgumentException: when applicationContext or userId is null*.