ChipletSDK接口文档

版本信息

- iOS13.0及以上版本
- 版本：1.0

业务开发配置

- 导入**ChipletSDK,**
- 打开**info.plist**文件，添加蓝牙权限描述，**Privacy-Bluetooth Always Usage Description**，填写Bluetooth使用描述。

- 确认Target->Generel->Framworks,Libraries**,.....->选中 Embed&Sign,**

SDK调用说明

- 引入头文件 **#import <ChipletSDK/ChipletSDK.h>**
- SDK提供五个回调方法，分别用于处理设备连接、指令超时、搜索设备列表、数据处理及数据库操作。

方法名	类型	描述

onConnectionChanged	void (^ _Nullable onConnectionChanged)(enumConnectionState, DeviceInfo * _Nullable)	处理设备连接断开状态
onCommandTimeout	void (^ _Nullable onCommandTimeout)(void)	指令是否超时，删除数据操作指令超时为8秒，其它指令超时为2秒
onDevicesDiscovered	void (^ _Nullable onDevicesDiscovered)(NSArray<DeviceInfo *> * _Nonnull)	搜索到的设备
onDataReceived	void (^ _Nullable onDataReceived)(NSDictionary<NSString *, id> * _Nonnull)	结果返回
onOperationHandler	void (^ _Nullable onOperationHandler)(enumOperateType, enumOperationStatus)	数据库操作结果返回

- bShowLog = true; 则显示打印信息
- deviceUUID 设置有值 ，则扫描到设备后，直接进行连接。

各类枚举值定义

```swift
@objc
public enum ReturnType : Int {
    case Battery                // 电量
    case ChargingStatus         // 充电状态
    case AppVersion             // 软件版本
    case FirmwareVersion        // 固件版本
    case SyncTime               // 同步时间
    case ReadTime               // 读取时间
    case HisData                // 所有记录
    case NoUploadedData         // 未上传记录
    case CleanData              // 清空记录
    case HeartRate              // 实时心率
    case Temperature            // 体温
    case O2                     // 血氧
    case Hrv                    // 实时心率变异性
    case StartOTA               // 开始OTA
    case EndOTA                 // OTA结束
    case Reset                  // 恢复出厂设置
    case AcquisitionCycle       // 采集周期
    case NegativeResponse       // 消极响应
    case Busy                   // 忙的提示
    case Progress               // 测量进度
}

@objc
public enum OperationStatus : Int {
    case OperationSuccessful        // 操作成功
    case OperationFailed            // 操作失败
}

@objc
public enum NegativeResponseType : Int {
    case TypeMeasuringInProgress    // 正在测量中
    case TypeUploadingHistory       // 正在上传历史记录
    case TypeDeletingHistory        // 正在删除历史记录
    case TypeOther                  // 其它
}

@objc
public enum ChargingType : Int {
    case TypeUncharged  // 未充电
    case TypeCharging   // 充电中
    case TypeFull       // 已充满
    case TypeOther      // 其它
}
```

```swift

@objc
public enum RingErrorType : Int {
    case TypeBusy           // 设备忙
    case TypeFailed         // 测量失败
    case TypeCharging       // 充电中
    case TypeNoWear         // 未配戴戒指
    case TypeInvalidData    // 数据非法
    case TypeNoNewData      // 没有新数据
}

// 定义连接状态的枚举
@objc
public enum ConnectionState : Int {
    case connected
    case disconnected
}

@objc
public enum OperateType : Int {
    case GetLastObject              // 获取最新一条数据
    case GetSleepObjectsOfDate      // 获取某一天的睡眠时间段数据
    case GetObjectsFromTimestamp    // 获取某一段时间的数据
    case GetObjectsOfDate           // 获取某一天的数据
    case DeleteAll                  // 删除所有数据
    case DeleteAllBeforeTimestamp   // 删除某时间段的数据
}
```

获取电量

调用 **readBatteryLevel** 获取电量。

请求参数

无

请求示例

获取电量 Objective-C

```objective-c
[BleTool.shared readBatteryLevel];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
        case ReturnTypeBattery: {
            // 读取电量  101 代码在充电中
            NSLog(@"当前电量: %ld", [message[@"value"] integerValue]);
        }
            break;
    }
};
```

注意事项

> 💡 数据统一放在onDataReceived进行处理，

充电状态

调用 **obtainChargingStatus** 获取充电状态。

请求参数

无

请求示例

充电状态 Objective-C

```
[BleTool.shared obtainChargingStatus];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
        case ReturnTypeChargingStatus: {
                // 充电状态
                ChargingType cType = [message[@"value"] integerValue];
                switch (cType) {
                    case ChargingTypeTypeUncharged: {
                        NSLog(@"充电状态: 未充电");
                    }
                        break;

                    case ChargingTypeTypeCharging: {
                        NSLog(@"充电状态: 充电中");
                    }
                        break;

                    case ChargingTypeTypeFull: {
                        NSLog(@"充电状态: 已充满");
                    }
                        break;

                    default:
                        NSLog(@"充电状态: 其它");
                        break;
                }
            }
                break;
    }
};
```

软件版本

- 调用getAppVersion获取软件版本

请求参数

无

请求示例

```Objective-C
[BleTool.shared getAppVersion];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeAppVersion: {
          // 软件版本
          NSLog(@"软件版本：%@", message[@"value"]);
      }
      break;
    }
};
```

固件版本

- 调用getFirmwareVersion读取固件版本

请求参数

无

请求示例

```Objective-C
[BleTool.shared getFirmwareVersion];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeFirmwareVersion: {
            // 固件版本
            NSLog(@"固件版本: %@", message[@"value"]);
        }
            break;
    }
};
```

同步时间

- 调用synchronizeTime进行同步时间

请求参数

无

请求示例

```objective-c
[BleTool.shared synchronizeTime];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeSyncTime: {
            // 同步时间
            OperationStatus opStatus = [message[@"value"] integerValue];
            NSLog(@"同步时间: %ld", opStatus);

            switch (opStatus) {
                case OperationStatusOperationSuccessful: {
                    NSLog(@"同步时间成功");
                }
                    break;

                case OperationStatusOperationFailed: {
                    NSLog(@"同步时间失败");
                }
                    break;

                default:
                    break;
            }
        }
            break;
    }
};
```

读取时间

- 调用readTime读取设备时间

请求参数

无

请求示例

```objective-c
[BleTool.shared readTime];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeReadTime: {
            // 读取时间
            NSLog(@"读取时间: %ld", [message[@"value"] integerValue]);

            // 使用NSDate类将时间戳转换为时间
            NSDate *date = [NSDate dateWithTimeIntervalSince1970:[message[@"value"] integerValue]/1000];

            // 使用NSDateFormatter将NSDate对象格式化为字符串
            NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
            [dateFormatter setDateFormat:@"yyyy-MM-dd HH:mm:ss"];
            NSString *formattedDate = [dateFormatter stringFromDate:date];

            NSLog(@"当前时间: %ld", formattedDate);
        }
            break;
    }
};
```

测量体温

- 调用measureTemperature测量体温

请求参数

无

请求示例

```objective-c
[BleTool.shared measureTemperature];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeTemperature: {
            // 测量体温
            NSLog(@"测量体温: %.02f", [message[@"value"] floatValue]);
        }
            break;
    }
};
```

测量实时心率变异性

- 调用measureHrv测量实时心率变异性

请求参数

无

请求示例

```objective-c
[BleTool.shared measureHrv];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeProgress: {
            // 测量进度
            NSLog(@"测量进度：%.02f", [message[@"value"] floatValue]);
        }
            break;

        case ReturnTypeBusy: {
            // 设备忙
            RingErrorType errorType = [message[@"value"] integerValue];

            switch (errorType) {
                case RingErrorTypeTypeBusy: {
                    NSLog(@"设备忙");
                }
                    break;

                case RingErrorTypeTypeFailed: {
                    NSLog(@"测量失败");
                }
                    break;

                case RingErrorTypeTypeCharging: {
                    NSLog(@"充电中");
                }
                    break;

                case RingErrorTypeTypeNoWear: {
                    NSLog(@"未配戴戒指");
                }
                    break;

                case RingErrorTypeTypeInvalidData: {
                    NSLog(@"数据非法");
                }
                    break;

                default:
                    break;
```

```
            }
        }
            break;

        case ReturnTypeHrv: {
            // 实时心率变异性
            NSLog(@"实时心率变异性: %.02f", [message[@"value"] floatValue]);
        }
            break;
    }
};
```

测量血氧

- 调用measureO2血氧

请求参数

无

请求示例

Objective-C

```
[BleTool.shared measureO2];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeProgress: {
            // 测量进度
            NSLog(@"测量进度：%.02f", [message[@"value"] floatValue]);
        }
            break;

        case ReturnTypeBusy: {
            // 设备忙
            RingErrorType errorType = [message[@"value"] integerValue];

            switch (errorType) {
                case RingErrorTypeTypeBusy: {
                    NSLog(@"设备忙");
                }
                    break;

                case RingErrorTypeTypeFailed: {
                    NSLog(@"测量失败");
                }
                    break;

                case RingErrorTypeTypeCharging: {
                    NSLog(@"充电中");
                }
                    break;

                case RingErrorTypeTypeNoWear: {
                    NSLog(@"未配戴戒指");
                }
                    break;

                case RingErrorTypeTypeInvalidData: {
                    NSLog(@"数据非法");
                }
                    break;

                default:
                    break;
```

```
            }
        }
            break;

        case ReturnTypeO2: {
            // 血氧
            NSLog(@"血氧: %.02f", [message[@"value"] floatValue]);
        }
            break;
    }
};
```

测量实时心率

- 调用measureHeartRate测量实时心率

请求参数

无

请求示例

Objective-C

```
[BleTool.shared measureHeartRate];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeProgress: {
            // 测量进度
            NSLog(@"测量进度：%.02f", [message[@"value"] floatValue]);
        }
            break;

        case ReturnTypeBusy: {
            // 设备忙
            RingErrorType errorType = [message[@"value"] integerValue];

            switch (errorType) {
                case RingErrorTypeTypeBusy: {
                    NSLog(@"设备忙");
                }
                    break;

                case RingErrorTypeTypeFailed: {
                    NSLog(@"测量失败");
                }
                    break;

                case RingErrorTypeTypeCharging: {
                    NSLog(@"充电中");
                }
                    break;

                case RingErrorTypeTypeNoWear: {
                    NSLog(@"未配戴戒指");
                }
                    break;

                case RingErrorTypeTypeInvalidData: {
                    NSLog(@"数据非法");
                }
                    break;

                default:
                    break;
```

```
            }
        }
            break;

        case ReturnTypeHeartRate: {
            // 实时心率
            NSLog(@"实时心率: %.02f", [message[@"value"] floatValue]);
        }
            break;
    }
};
```

同步新记录

- 调用getNoUpdateRecords同步新数据到数据库

请求参数

无

请求示例

```objective-c
[BleTool.shared getNoUpdateRecords];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeNoUploadedData: {
            if ([message[@"value"] isKindOfClass:[NSDictionary class]]) {
                NSLog(@"同步进度：%.02f", [message[@"value"][@"progress"] floatValue]);
            }else if ([message[@"value"] isKindOfClass:[NSNumber class]]) {
                NSInteger opStatus = [message[@"value"] integerValue];

                switch (opStatus) {
                    case OperationStatusOperationSuccessful: {
                        NSLog(@"同步成功");
                    }
                        break;

                    case RingErrorTypeTypeNoNewData: {
                        NSLog(@"暂无新数据");
                    }
                        break;

                    default:
                        break;
                }
            }else {
                NSLog(@"不应该存在这种情况");
            }
        }
            break;
    }
};
```

所有记录

- 调用getAllRecords同步所有记录

请求参数

无

请求示例

```Objective-C
[BleTool.shared getAllRecords];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeHisData: {
            if ([message[@"value"] isKindOfClass:[NSDictionary class]]) {
                NSLog(@"同步进度：%.02f", [message[@"value"][@"progress"] floatValue]);
            }else if ([message[@"value"] isKindOfClass:[NSNumber class]]) {
                NSInteger opStatus = [message[@"value"] integerValue];

                switch (opStatus) {
                    case OperationStatusOperationSuccessful: {
                        NSLog(@"同步成功");
                    }
                        break;

                    case RingErrorTypeTypeNoNewData: {
                        NSLog(@"暂无数据");
                    }
                        break;

                    default:
                        break;
                }
            }else {
                NSLog(@"不应该存在这种情况");
            }
        }
            break;
    }
};
```

清空本地记录

- 调用cleanHisData清空本地记录

请求参数

无

请求示例

```Objective-C
[BleTool.shared cleanHisData];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeCleanData: {
            OperationStatus opStatus = [message[@"value"] integerValue];

            switch (opStatus) {
                case OperationStatusOperationSuccessful: {
                    NSLog(@"清空成功");
                }
                    break;

                case OperationStatusOperationFailed: {
                    NSLog(@"清空失败");
                }
                    break;

                default:
                    break;
            }
        }
            break;
    }
};
```

设置采集周期

- 调用setAcquisitionCycleWithText:设置采集周期

请求参数

参数	类型	描述
text	NSString	采集周期的值，单位秒

请求示例

```Objective-C
// 方便输入框直接给值，单位 秒
[BleTool.shared setAcquisitionCycleWithText:@"60"];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeAcquisitionCycle: {
            OperationStatus opStatus = [message[@"value"] integerValue];

            switch (opStatus) {
                case OperationStatusOperationSuccessful: {
                    NSLog(@"设置成功");
                }
                    break;

                case OperationStatusOperationFailed: {
                    NSLog(@"设置失败");
                }
                    break;

                default:
                    break;
            }
        }
            break;
    }
};
```

💡 方便输入框直接给值，单位 秒。最小值为60,小于60，则会回报失败。

恢复出厂设置

- 调用reset恢复出厂设置

请求参数

无

请求示例

```Objective-C
[BleTool.shared reset];

BleTool.shared.onDataReceived = ^(NSDictionary<NSString *,id> * _Nonnull message) {
    NSLog(@"收到消息：%@", message);
    ReturnType type = [message[@"type"] integerValue];
    switch (type) {
      case ReturnTypeReset: {
            OperationStatus opStatus = [message[@"value"] integerValue];

            switch (opStatus) {
                case OperationStatusOperationSuccessful: {
                    NSLog(@"恢复出厂设置成功");
                }
                    break;

                case OperationStatusOperationFailed: {
                    NSLog(@"恢复出厂设置失败");
                }
                    break;

                default:
                    break;
            }
        }
            break;
    }
};
```

数据格式描述

```swift
public override var description: String {
    return """
    时间: \(self.timestamp)
    当天累计步数: \(self.stepsOfTheDay)
    心率: \(self.rate)
    血氧: \(self.O2)
    心率变异性: \(self.hrv)
    压力指数: \(self.mentalStress)
    温度: \(self.temp)
    是否运动: \(self.isRunning)
    睡眠类型: \(self.sleepType)
    """
}
```

获取最新记录

- 调用getLatestObject获取最新一条记录

请求参数

无

请求示例

```objective-c
LocalDataModel *dataModel = [DBTool.shared getLatestObject];


```

获取指定日期的睡眠时间段数据

- 调用getSleepObjectsOf:获取指定日期的睡眠时间段数据

请求参数

参数	类型	描述

date	NSDate	指定日期

请求示例

```Objective-C
NSArray<LocalDataModel *> *arr = [DBTool.shared getSleepObjectsOf:date];

```

获取指定日期的数据

- 调用**getObjectsOf**:获取指定日期的数据

请求参数

参数	类型	描述
date	NSDate	指定日期

请求示例

```Objective-C
NSArray<LocalDataModel *> *arr = [DBTool.shared getObjectsOf:date];

```

获取指定时间段的数据

- 调用**getObjectsFrom**:获取指定时间以来的数据

请求参数

参数	类型	描述
timestamp	NSTimeInterval	指定时间的时间戳

请求示例

```Objective-C
NSArray<LocalDataModel *> *arr = [DBTool.shared getObjectsFrom:timestamp];


```

删除指定时间以前的数据

- 调用deleteAllBeforeTimestampWithTimestamp:获取指定时间以来的数据

请求参数

参数	类型	描述
timestamp	NSTimeInterval	指定时间的时间戳

请求示例

```Objective-C
[DBTool.shared deleteAllBeforeTimestampWithTimestamp:timestamp];

DBTool.shared.onOperationHandler = ^(enum OperateType opType, enum OperationStatus opStatus) {
    switch (opType) {
        case OperateTypeDeleteAllBeforeTimestamp: {
            if (opStatus == OperationStatusOperationSuccessful) {
                [XNProgressHUD.shared showSuccessWithTitle:@"删除成功"];
            }else {
                [XNProgressHUD.shared showErrorWithTitle:@"删除失败"];
            }
        }
            break;

        default:
            break;
    }
};
```

删除全部数据

- 调用deleteAll:获取指定时间以来的数据

请求参数

无

请求示例

```objective-c
[DBTool.shared deleteAll];

DBTool.shared.onOperationHandler = ^(enum OperateType opType, enum OperationStatus opStatus) {
    switch (opType) {
        case OperateTypeDeleteAll: {
            if (opStatus == OperationStatusOperationSuccessful) {
                [XNProgressHUD.shared showSuccessWithTitle:@"删除成功"];
            }else {
                [XNProgressHUD.shared showErrorWithTitle:@"删除失败"];
            }
        }
            break;

        default:
            break;
    }
};
```