AsReader

AsReader ASK SDK

AsReader ASK SDK Reference Guide V1.1

AsReader, Inc.

Content

1. JvmaSDK Class

トップレベル（アプリケーションレベル）のインターフェースクラスになります。アプリケーションに JVMA のインターフェースを提供します。

1.1 initWithAsReaderAsk

Syntax：

```
-(instancetype)initWithAsReaderAsk:(AsReaderAsk*)asReaderAsk;
```

Remarks：
インスタンス初期化メソッドです。AsReaderAsk インスタンスをパラメータとして渡す必要があります。AsReaderAsk は後述にありますが、ASK 通信を実現するモジュールのインターフェースを提供するクラスになります。

1.2 sendTestModeResult

Syntax：

```
- (void)sendTestModeResult:(TestModeBlock)result;
```

Remarks：
テストモードを設定します。

1.3 sendTestDatasWithJvmaDatas

Syntax：

```
- (void)sendTestDatasWithJvmaDatas:(NSArray<JvmaRecord*> *)jvmaDatas
resultDataBlock:(ResultDataBlock)resultDataBlock;
```

Remarks：
テストデータを送信し、JvmaRecord のインスタンスの配列及びブロックを引き渡します。

1.4 startCommunicationWithSettingCode

Syntax：

```
-(JvmaResult*)startCommunicationWithSettingCode:(NSString *)settingCode
TerminalPassword:(NSString *)terminalPassword
resultDataBlock:(ResultDataBlock)resultDataBlock;
```

Remarks：
汎用交信開始メソッド、文字列のパスワードを渡して交信開始プロセスを行います。交信結果は戻り値で確認を行う。

1.5 sendCommandWithSettingCode

Syntax：

```
-(JvmaResult*)sendCommandWithSettingCode:(NSString*)settingCode
terminalPassword:(NSString*)terminalPassword commandCode:(CommandCode)commandCode
dataIds:(NSArray<JvmaDataId*> *)dataIds inputTimeRecord:(BOOL)inputTimeRecord
resultDataBlock:(ResultDataBlock)resultDataBlock;
```

Remarks：
コマンド送信用のメソッドです。交信コマンドレコードの識別コードと識別コード指定レコードで指定するデータ識別コードの配列を渡して処理します。データ識別コードの配列は必須ではなりません。
引数はSettingCode、terminalPassword、CommandCode（交信コマンド）、dataIds（識別コード配列）、inputTimeRecord（タイムの情報入力の有無）、戻り値はJvmaResultのインスタンスです。

1.6 getVmDataWithSettingCode

Syntax：

```
-(JvmaResult*)getVmDataWithSettingCode:(NSString *)settingCode
TerminalPassword:(NSString*)terminalPassword dataIds:(NSArray<JvmaDataId*>*)dataIds
inputTimeRecord:(BOOL)inputTimeRecord resultDataBlock:(ResultDataBlock)resultDataBlock;
```

Remarks：
自販機データ収集交信メソッドです。識別コード指定レコードで指定するデータ識別コードの配列を渡して、特定のレコードを取得することもできます。引数は SettingCode と terminalPassword、dataIds(識別コード配列)、inputTimeRecord(タイムの情報入力の有無)、戻り値は JvmaResult のインスタンスです。

1.7 getVmDataClearWithSettingCode

Syntax：

```
- (void)getVmDataClearWithSettingCode:(NSString *)settingCode terminalPassword:(NSString
*)terminalPassword dataIds:(NSArray<JvmaDataId*> *)dataIds
inputTimeRecord:(BOOL)inputTimeRecord resultDataBlock:(ResultDataBlock)resultDataBlock
clearResultDataBlock:(ResultDataBlock)clearResultDataBlock;
```

Remarks：
自販機データクリア交信のメソッドです。引数は SettingCode と terminalPassword、dataIds(識別コード配列)、inputTimeRecord(タイムの情報入力の有無)、戻り値は JvmaResult のインスタンスです。

1.8 collectSettingDataWithSettingCode

Syntax :

```
-(JvmaResult*)collectSettingDataWithSettingCode:(NSString *)settingCode
TerminalPassword:(NSString*)terminalPassword inputTimeRecord:(BOOL)inputTimeRecord
resultDataBlock:(ResultDataBlock)resultDataBlock;
```

Remarks :
設定データ収集交信のメソッドです。引数は SettingCode と terminalPassword、inputTimeRecord(タイムの情報入力の有無)、戻り値は JvmaResult のインスタンスです。

1.9 sendSettingDataWithSettingCode

Syntax :

```
-(JvmaResult*)sendSettingDataWithSettingCode:(NSString *)settingCode
TerminalPassword:(NSString*)terminalPassword
settingDatas:(NSArray<JvmaRecord*>*)settingDatas inputTimeRecord:(BOOL)inputTimeRecord
resultDataBlock:(ResultDataBlock)resultDataBlock;
```

Remarks :
設定データ設定交信のメソッドです。引数は SettingCode と terminalPassword、JvmaRecord、inputTimeRecord(タイムの情報入力の有無)、戻り値は JvmaResult のインスタンスです。

1.10 allClearWithSettingCode

Syntax :

```
- (void)allClearWithSettingCode:(NSString *)settingCode
terminalPassword:(NSString *)terminalPassword
inputTimeRecord:(BOOL)inputTimeRecord
clearResultDataBlock:(ResultDataBlock)clearResultDataBlock;
```

Remarks :
データオールクリア交信のメソッドです。引数は SettingCode、terminalPassword、inputTimeRecord(タイムの情報入力の有無)、戻り値は JvmaResult のインスタンスです。

1.11 sendJvmaDatasWithSettingCode

Syntax：

```
-(JvmaResult*)sendJvmaDatasWithSettingCode:(NSString*)settingCode
        terminalPassword:(NSString*)terminalPassword
        jvmaDatas:(NSArray<JvmaRecord*>*)jvmaDatas
resultDataBlock:(ResultDataBlock)resultDataBlock;
```

Remarks：
データ送信用の汎用共通メソッドです。JVMA レコードを渡して、ASK モジュール経由で送信します。引数は SettingCode、terminalPassword、JvmaRecord、戻り値は JvmaResult のインスタンスです。

1.12 getJvmaModuleVersionResultVersionBlock

Syntax：

```
-(void)getJvmaModuleVersionResultVersionBlock:(ResultVersionBlock)r
esultVersionBlock;
```

Remarks：
ASK 通信モジュールのバージョンを取得する。

1.13 setSDKLogOn

Syntax：

```
- (void)setSDKLogOn:(BOOL)on;
```

Remarks：
ログを設置する。

1.14 removeAllSdkLog

Syntax：

```
- (void)removeAllSdkLog;
```

Remarks：
すべてのログを削除する。

1.15 getVersion

Syntax：

```
- (NSString *)getVersion;
```

Remarks：
バージョン番号を取得する。

1.16 sendSettingDataWithSettingCode

Syntax：

```
- (void)sendSettingDataWithSettingCode:(NSString *)settingCode
terminalPassword:(NSString *)terminalPassword
settingDatas:(NSArray<JvmaRecord*> *)settingDatas
commandCode:(CommandCode)commandCode inputTimeRecord:(BOOL)inputTimeRecord
resultDataBlock:(ResultDataBlock)resultDataBlock;
```

Remarks：
設定データ設定交信のメソッドです。引数は SettingCode、terminalPassword、JvmaRecord、commandCode 、inputTimeRecord(タイムの情報入力の有無)、戻り値は JvmaResult のインスタンスです。

1.17 collectSettingDataWithSettingCode

Syntax：

```
- (void)collectSettingDataWithSettingCode:(NSString *)settingCode
terminalPassword:(NSString *)terminalPassword
commandCode:(CommandCode)commandCode inputTimeRecord:(BOOL)inputTimeRecord
resultDataBlock:(ResultDataBlock)resultDataBlock;
```

Remarks：
設定データ収集交信のメソッドです。引数は SettingCode、terminalPassword、commandCode、inputTimeRecord(タイムの情報入力の有無)、戻り値は JvmaResult のインスタンスです。

1.18 setTimeout:(float)second

Syntax：

```
- (void)setTimeout:(float)second;
```

Remarks：

SDK のタイムアウト時間を設定するメソッドです（このメソッドを呼び出さない場合、タイムアウト時間が 10 秒です）。引数の設定範囲は６秒以上です。引数を０に設定する場合、SDK がタイムアウトを処理しません。引数は０以上、６以内に設定する場合、６として設定されます。

2. JvmaRecord Class

JVMA レコードを実装したクラスです。汎用送信メソッドでレコードを送信したい場合、もしくは JvmaResult から結果を解析したい時にこのクラスを使います。

2.1 Properties

```
@property (nonatomic ,strong, readonly) JvmaDataId *jvmaDataId;
 //JvmaDataIdのインスタンス
```

```
@property (nonatomic ,strong, readonly) NSString *recordLengthBCD;
//BCDフォーマットのデータの長さ
```

```
@property (nonatomic ,strong, readonly) NSString *itemDigitsBCD;
// BCDフォーマットのitemデータの桁数
```

```
@property (nonatomic ,strong, readonly) NSArray *items;
//JvmaItem 配列
```

```
@property (nonatomic, strong, readonly) NSString *hexStr;
//１６進数フォーマットのデータ
```

```
@property (nonatomic, strong, readonly) NSData *data;
// Data フォーマットのデータ
```

```
@property (nonatomic, assign, readonly) NSUInteger bytes;
// NSUInteger フォーマットのデータ
```

2.2 initWithData

Syntax :

```
- (instancetype)initWithData:(NSData *)data;
```

Remarks :
バイナリデータからインスタンスを生成するメソッドです。

2.3 initWithJvmaDataIdCode

Syntax :

```
-  (instancetype)initWithJvmaDataIdCode:(JvmaDataIdCode)JvmaDataIdCode
   items:(NSArray<JvmaItem*> *)items;
```

Remarks :
識別コードとアイテムを渡してレコードインスタンスを生成します。

3. JvmaItem Class

JVMA レコードのアイテムを実装するクラスです。レコードをインスタンス化またはレコード内容を解析する時に使うクラスです。

3.1 Properties

```
@property (nonatomic, strong, readonly) NSString *hexStr;
//１６進数フォーマットのデータ
```

```
@property (nonatomic, strong, readonly) NSString *digitsBCD;
//BCD フォーマットのデータ桁数
```

```
@property (nonatomic, strong, readonly) NSUInteger digits;
//データ桁数
```

```
@property (nonatomic, strong, readonly) NSUInteger bytes;
//データの長さ
```

```
@property (nonatomic, strong, readonly) NSString *byteBCD;
//BCD フォーマットバイト
```

3.2 initWithData

Syntax：

```
- (instancetype)initWithData:(NSData *)data;
```

Remarks：

初期化、NSData 対象を引き渡します。

3.3 initWithHexStr

Syntax：

```
- (instancetype)initWithHexStr:(NSString *)hexStr;
```

Remarks：

初期化、１６進数のストリングを引き渡します。

4. JvmaResult Class

JVMA 交信結果を定義するクラスです。

4.1 property

```
@property (nonatomic ,strong ,readonly) JvmaError *error;
// エラー内容、正常の場合は Nil
```

```
@property (nonatomic ,strong ,readonly) JvmaACKData *ack;
// JvmaACKData のオブジェクト
```

```
@property (nonatomic ,strong ,readonly) JvmaNAKData *nak;
// JvmaNAKData のオブジェクト
```

```
@property (nonatomic ,strong ,readonly) JvmaNAKData *nak;
// JvmaNAKData のオブジェクト
```

```
@property (nonatomic ,strong ,readonly) NSMutableArray<JvmaRecord*> *jvmaTexts;
// JvmaRecordのオブジェクトの配列を保存
```

4.2 initWithOfflinePassword

Syntax :

```
- (instancetype)initWithOfflinePassword:(NSString
*)offlinePassword TerminalPassword:(NSString *)terminalPassword;
```

Remarks :

オフラインパスワードの初期化です。

4.3 initWithErrorCode

Syntax :

```
- (instancetype)initWithErrorCode:(int)errorCode;
```

Remarks :

errorCode を引き渡します。

4.4 addResultData

Syntax：

```
- (BOOL)addResultData:(NSData *)resultData;
```

Remarks：
Data フォマートのデータを追加します。

4.5 getAllRecords

Syntax：

```
- (NSArray<JvmaRecord*> *)getAllRecords;
```

Remarks：
結果セット内のレコードを全部取得します。

4.6 getRecordWithDataId

Syntax：

```
- (JvmaRecord*)getRecordWithDataId:(JvmaDataIdCode)dataId;
```

Remarks：
結果セット内のレコードを取得します。JvmaDataIdCode インスタンスを引き渡します。

5. JvmaError Class

交信エラー内容を定義するクラスです。交信結果のメンバーとして存在します。

5.1 Properties

```
@property (nonatomic, strong, readonly)NSString *errorCode;
//エラーコード
//JvmaError のインスタンス
```

```
@property (nonatomic, strong, readonly)NSString *errorMessage;
//エラー内容
//JvmaError のインスタンス
```

5.2 initWithErrorCode

Syntax :

```
- (instancetype)initWithErrorCode:(int)errorCode;
```

Remarks :
初期化、errorCode を引き渡します。

6. JvmaDataId Class

JVMA レコードの識別コードを実装するクラスです。

Typedef enum

```
typedef NS_ENUM(NSInteger, JvmaDataIdCode){
    VmIdentifyCode          = 0x0A1A, //識別コード指定レコード（HTからVMへの転送データ）
    CommunicationCommand    = 0x0AA0, //交信コマンドレコード（HTからVMへの転送データ）
    ClearStutas             = 0x0AA1, //クリア状態データレコード（VMからHTへの転送データ）
    VmTime                  = 0x0BA0, //時刻レコード（HTからVMへの転送データ）
    ManageData              = 0x0BA1, //自販機管理データレコード（HTからVMへ、VMからHTへ
                                      の転送データ）
    IndividualData          = 0x0BB0, //自販機個別データレコード（VMからHTへの転送データ）
    VmFaultCode             = 0x0BB1, //故障コード履歴レコード（VMからHTへの転送データ）
    MonitorCode             = 0x0BB2, //モニターコード履歴レコード（VMからHTへの転送
                                      データ）
    OfflineLog              = 0x0BB3, //オフライン用交信LOGレコード（VMからHTへの転送
                                      データ）
    SetTotal                = 0x0BB4, //トータル集計レコード（VMからHTへの転送データ）
    SetTotalBackUp          = 0x0BB5, //トータル集計バックアップレコードVMからHTへの転送
                                      データ）
    PriceQuantity           = 0x0BB6, //価格別累計販売数レコード（VMからHTへの転送データ）
    DateQuantity            = 0x0BB7, //日別販売数レコード（VMからHTへの転送データ）
    MaterialKindCode        = 0x0BB8, //商材別コードレコード（カップ機）
    SupMaterialKindCode     = 0x0BB9, //補_材別コードレコードカップ機）
    SaleAmount              = 0x0BBA, //部署別IDカード販売金額レコード(VMからHTへの転送
                                      データ）
    SaleAmountBackUp        = 0x0BBB, //部署別 ID カード販売金額バックアップレコード
    ShouhinCD               = 0x0BC0, //商品コードレコード（HTからVMへ、VMからHTへの転送
                                      データ）
    KindCD                  = 0x0BC1, //分類コードレコード（パッケージ商品機）
    CupKindCD               = 0x0BC2, //分類コードレコード（カップ機）(VMからHTへの転送
                                      データ）
    ColumnCashPrice         = 0x0BC3, //コラム別現金売り販売単価レコード
    ColumnQuantity          = 0x0BC4, //コラム別販売数レコード(VMからHTへの転送データ）
    ColumnSaleOutTime       = 0x0BC5, //コラム別売切時間レコード
    ColumnCardPrice         = 0x0BC6, //コラム別カード販売単価レコード
    PrepayCardQuantity      = 0x0BC7, //コラム別プリペイドカード売り販売数レコード
    PrepayCardSaleAmount    = 0x0BC8, //コラム別プリペイドカード･現金併用販売時カード売り
                                      金額レコード(VMからHTへの転送データ）
    IdCardQuantity          = 0x0BC9, //コラム別IDカード売り販売数レコード
    FreeSaleQuantity        = 0x0BCA, //コラム別フリーベンド販売数レコード
    InputCashQuantity       = 0x0BE0, //金種別投入枚数レコード(VMからHTへの転送データ）
    QuitCashQuantity        = 0x0BE1, //金種別排出枚数レコード(VMからHTへの転送データ）
    ChangEmptyTime          = 0x0BE2, //釣切時間レコード(VMからHTへの転送データ）
```

```
VmRunTime                = 0x0BE3, //自販機稼働時間レコード(VMからHTへの転送データ)
    MonitorSaleOutTime       = 0x0BE4, //売切モニター時間レコード
    MaterialSaleOutTime      = 0x0BE5, //商材別売切時間レコード(VMからHTへの転送データ)
    SupMaterialSaleOutTime   = 0x0BE6, //補材別売切時間レコード(VM から HT への転送データ)
    LargeValueInputQuantity = 0x0BE7, //高額金種別投入枚数レコード
    LargeValueQuitQuantity   = 0x0BE8, //高額金種別払出枚数レコード
    LargeValueChangEmptyTime= 0x0BE9, //高額金種別釣切時間レコード
    Total                    = 0x0C0A, //トータル累計レコード(VM から HT への転送データ)
    TotalSaleQuantity        = 0x0C0B, //コラム別累計販売数レコード
    OfflinePassword          = 0x0D0B, //オフラインパスワード
    TerminalPassword         = 0x0D0D  //ターミナルパスワード
    CenterPasswordRecord     = 0x0D0C, //センターパスワードレコード

    //自販機～ハンディターミナル間の交信仕様
#pragma mark F010-1 追加仕様版（2000円札･小額金種対応版）Ver.1.0.5
    InputMoneyQuantity       = 0x0BEA, //金種別投入枚数レコード２
    ExhaustMoneyQuantity     = 0x0BEB, //金種別排出枚数レコード２
    NoChangeTime             = 0x0BEC, //金種別釣切時間レコード２
#pragma mark F010-2 追加仕様版（チャージ金額対応）Ver.1.0.0
    TotalRecords             = 0x0BBC, //トータル集計レコード２
    DataBackupCD             = 0x0BBD, //トータル集計バックアップレコード２
    CumulativeRecord         = 0x0C7A, //トータル累計レコード２
#pragma mark F010-3 追加仕様版（マルチブランド対応版）Ver.1.0.0
    BrandNumberRecord        = 0x0BD0, //ブランド番号レコード
    BrandNameRecord          = 0x0BD1, //ブランド名称レコード
    AllBrandsRecord          = 0x0BD2, //ブランド別トータル集計レコード
    AllBrandsDataBackupCD    = 0x0BD3,//ブランド別トータル集計バックアップレコード
    DifferentBrandsSaleNum   = 0x0BD4, //ブランド別コラム別販売数
    DifferentBrandsSalePrice= 0x0BD5, //ブランド別コラム別販売価格（VMからHTへの転送
                                      データ）
    BrandSalaQuantity        = 0x0BD6, //ブランド別販売数（VMからHTへの転送データ）
   TotalBrandRecord          = 0x0BDA, //ブランド別トータル累計レコード
    OtherBrandSaleQuantity  = 0x0BDB, //ブランド別コラム別累計販売数
                                      (VMからHTへの転送データ)
TotalBrandQuantity      = 0x0BDC, //ブランド別累計販売数
                                      (VMからHTへの転送データ)
#pragma mark F010-4 追加仕様版（売上集計機能追加）Ver.1.0.0
    DayTotalRecord1          = 0x0BBE, //当日トータル集計レコード
    DayTotalRecord2          = 0x0BBF, //当日トータル集計レコード２
    DayTotalSales            = 0x0BCC, //日別売上金累計レコード
#pragma mark F010-5 追加仕様版（ルーレット対応版）Ver.1.0.0
    ColumnRouletteNumber     = 0x0BCD, //コラム別ルーレット当たり回数レコード
#pragma mark F010-6 追加仕様版（高額商品対応版）Ver.1.0.0
    HighPriceSaleNumRecord   = 0x0BEF, //高額商品対応価格別累計販売数レコード
    HighPriceOtherSaleRecord= 0x0BED, //高額商品対応コラム別現金売り販売単価レコード
```

```
#pragma mark F010-7 追加仕様版（販売単価設定追加）Ver.1.0.0
    AnotherCashSaleSetRecord= 0x0CC0, //コラム別現金売り販売単価設定レコード
    AnotherCardSaleSetRecord= 0x0CC1, //コラム別カード販売単価設定レコード
    BrandOtherSalePrice     = 0x0CC2, //ブランド別コラム別販売単価設定レコード
    CashSaleChangeRecord    = 0x0CC3, //現金売り販売単価変更履歴レコード
    CardSalePriceChange     = 0x0CC4, //カード売り販売単価変更履歴レコード
    BrandOtherSalePriceChange=0x0CC5  //ブランド別コラム別販売単価変更履歴レコード
                                      （VMからHTへの転送データ）
    VersionRecord          =0xFFFF,
    TestRecord             =0xEEEE,
    //補給コマンド
    SupplyCommand1 = 0x0A0A,        //交信モードデータ
    SupplyCommand2 = 0x0B1A,        //補給データ（バッケージ商品機）
    CenterIP            = 0x0C5A,
    VMIP               = 0x0C5B,
    //電話番号仕様：10（電話番号の桁数）、20桁の１アイテムを桁数(2桁)＋電話番号
(18桁)として可変長の有効電話番号です。
    TcpIPSendNo         = 0x0C5C,
    AdministrationGroupNo = 0x0B0A,
    VMCall              = 0x0B0C,
};
```

6.1 Property

```
@property (nonatomic, assign ,readonly) JvmaDataIdCode code;
//JvmaDataIdCode 列挙型の識別コード
```

```
@property (nonatomic, assign ,readonly) NSString *hexStrCode;
//１６進数のヘッダーのストリング
```

6.2 initWithCode

Syntax：

```
- (instancetype)initWithCode:(JvmaDataIdCode)code;
```

Remarks：
JvmaDataIdCode の列挙型の識別コードで初期化します。

6.3 initWithHexStrCode

Syntax：

```
-(instancetype)initWithHexStrCode:(NSString *)hexStrCode;
```

Remarks：
１６進数のコードで初期化します。

6.4 initWithHexData

Syntax：

```
- (instancetype)initWithHexData:(NSData *)hexData;
```

Remarks：
データで初期化します。

7. AsReaderAsk Class

ASK 通信モジュールを実装したクラスです。モジュールのコントロールをするインターフェースを提供します。

7.1 AsReaderAskStatusDelegate

```
@protocol AsReaderAskStatusDelegate <NSObject>
//デリゲートプロトコル
```

7.1.1 onReceivedConnectedState

Syntax：

```
- (void)onReceivedConnectedState:(BOOL)isConnected;
```

Remarks：
AsReaderASK が iOS デバイスと接続した時に呼び出されるデリゲートメソッドです。

7.1.2 onReceivedBatterySate

Syntax：

```
- (void)onReceivedBatterySate:(int)gauge;
```

Remarks：
バッテリーの電量の通知を受け取るメソッドです。

7.1.3 onReceivedTriggerState

Syntax：

```
- (void)onReceivedTriggerState:(BOOL)isTriggerDown;
```

Remarks：
トリガーボタンのイベントを受け取るメソッドです。ステータス：1 は押下、0 は解放です。

7.1.4 onReceivedPowerState

Syntax：

```
- (void)onReceivedPowerState:(BOOL)isOn beep:(BOOL)isBeep vib:(BOOL)isVib
led:(BOOL)isLed;
```

Remarks：
電源、音声、振動、LED ランプの状態をリータンします。

7.1.5 onReceivedReaderSettingBeep

Syntax：

```
- (void)onReceivedReaderSettingBeep:(BOOL)isOn Vib:(BOOL)isVib
Led:(BOOL)isLed;
```

Remarks：
音声、振動、LED ランプの状態をリータンします。

7.2 AsReaderAskDataDelegate

```
@protocol AsReaderAskDataDelegate <NSObject>
```

7.2.1 reciveData

Syntax：

```
- (void)reciveData:(NSData *)reciveData;
```

Remarks：
AsReaderASK がデータを受信するメソッドです。

7.3 Properties

```
@property (nonatomic, assign) BOOL isConnected;
 //接続状態
```

7.4 sharedAsReaderAsk

Syntax：

```
+ (AsReaderAsk *)sharedAsReaderAsk;
```

Remarks：
シングルトンオブジェクトを生成。

7.5 sendPowerOn

Syntax：

```
- (BOOL)open;
```

Remarks：
電源、音声、LED ランプ等を設定します。

7.6 sendSettingBeep

Syntax：

```
- (BOOL) sendSettingBeep:(BOOL)isBeep
setVib:(BOOL)isVib setLed:(BOOL)isLED;
```

Remarks：
音声、LED ランプ等を設定します。

7.7 setAsReaderAskLogOn

Syntax：

```
- (void)setAsReaderAskLogOn:(BOOL)on;
```

Remarks：
コンソールログ出力要否の設定を行うメソッドです。

7.8 setTriggerModeDefault

Syntax：

```
- (void)setTriggerModeDefault:(BOOL)isDefault;
```

Remarks：
トリガーの状態を捉えるかどうかの指定メソッドです。

7.9 sendData

Syntax：

```
- (BOOL)sendData:(NSData *)data;
```

Remarks：
送信メソッドです。

8. JvmaText Class

データテキストのクラスです。

typedef　enum

```
typedef NS_ENUM(NSInteger, JvmaHasNext){
    JvmaHasNextTypeETX        =0x03,
    JvmaHasNextTypeETB        =0x17
};//ETB がデータの途中のパケットの末尾とし、
//ETX がデータの最後のパケットの末尾する。
```

```
typedef NS_ENUM(NSInteger, JvmaTextType){
    JvmaTextTypePassword      =0x50, //パスワード
    JvmaTextTypeCommand       =0x43, //コマンド
    JvmaTextTypeData          =0x44, //データ
    JvmaTextTypeVersion       =0x56 //バージョン
    JvmaTextTypeTest          =0x54 //テストモード
};//text タイプ
```

```
typedef NS_ENUM(NSInteger, CommandCode){
    VmDataCollect           = 0x01, //自販機データ収集
    VmDataCollectClear       = 0x02, //自販機データ収集クリア
    SettingDataCollect       = 0x03, //設定データ収集
    SettingDataSet          = 0x41, //設定データ設定
    AllClear               = 0x81, //オールクリア
    VmDataCollectClearConfirm = 0x87, //０２クリア確認
    AllClearConfirm          = 0x88, //８１クリア確認

    OnLineSettingDataCollect  = 0x06,//T
    OnLineSettingDataSet     = 0x46,//T
    OnLineWorkingSettingDataSet=0x47,//T
    OnLineAllClear          = 0x86,//T
    OnLineAllClearConfirm    = 0x89,//T

    TCP_IPSettingDataCollect  = 0x08,//T
    TCP_IPSettingDataSet     = 0x48,//T

    CenterPasswordSet        = 0x92,//T
};
```

8.1 Property

```
@property (nonatomic ,assign, readonly) textHeader     stx;
//データのヘーダー、デフォルトが０２
```

```
@property (nonatomic ,assign,readonly) JvmaTextType textType;
// text タイプ
```

```
@property (nonatomic ,strong, readonly) NSString  *dataLengthBCD;
// BCD フォーマットのデータの長さ
```

```
@property (nonatomic ,strong)   NSNumber    *blockNumber;
// NSNumber 型のデータ
```

```
@property (nonatomic ,strong, readonly) NSString    *bbcHex;
// bbc１６進数
```

```
@property (nonatomic ,strong, readonly) NSArray *jvmaRecords;
// jvmaRecords のインスタンスを保存する配列
```

```
@property (nonatomic ,assign) JvmaHasNext  hasNext;
//ETB がデータの途中のパケットの末尾とし、
//ETX がデータの最後のパケットの末尾とします。
```

```
@property (nonatomic ,strong, readonly) NSString  *hexStr;
//  １６進数でのデータ
```

```
@property (nonatomic ,strong, readonly) NSData     *hexData;
// Data 型のデータ
```

```
@property (nonatomic ,strong, readonly) NSString *offlinePassword;
//オフラインパスワード
```

```
@property (nonatomic ,strong, readonly) NSString *terminalPassword;
//ターミナルパスワード
```

```
@property (nonatomic ,assign) BOOL isTestData;
//テストデータ
```

8.2 initWithOfflinePassword

Syntax：

```
- (instancetype)initWithOfflinePassword:(NSString
*)offlinePassword TerminalPassword:(NSString *)terminalPassword
textType:(JvmaTextType)textType;
```

Remarks：
初期化です。引数が offlinePassword、terminalPassword と text タイプです。

8.3 addjvmaRecord

Syntax：

```
- (BOOL)addjvmaRecord:(JvmaRecord *)jvmaRecord;
```

Remarks：
jvmaRecord データを追加します。戻り値が BOOL 型で、no の場合データ容量の最大限で、yes の場合、データを連続追加可能です。

8.4 initWithOfflinePassword

Syntax：

```
- (instancetype)initWithOfflinePassword:(NSString
*)offlinePassword TerminalPassword:(NSString *)terminalPassword
jvmaData:(NSData *)jvmaData;
```

Remarks：
初期化メソッドです。引数は offlinePassword、terminalPassword 及び data フォマートのデータです。

9. JvmaACKData Class

ACK データのクラスです。

typedef enum

```
typedef NS_ENUM(NSInteger, JvmaACKStatus) {
    VmSucceed = 0x00,//通信の肯定応答
    VmStarted = 0x01,//交信開始ステータス
    VmConfirmConnect = 0x02,//接続確立ステータス。
    VmCommandConfirm= 0x03 //コマンド確認ステータス。
};//ACK タイプ
```

9.1 Property

```
@property (nonatomic ,assign, readonly) textHeader ack;
//ACK ヘッダー
```

```
@property (nonatomic ,assign, readonly) JvmaACKStatus status;
//ACK タイプ
```

```
@property (nonatomic, strong, readonly) NSData   *hexData;
//ACK データ
```

```
@property (nonatomic, strong ,readonly) NSString *hexStatus;
//当面ステータスの 16 進数文字列
```

9.2 initWithStatus

Syntax：

```
- (instancetype)initWithStatus:(JvmaACKStatus)status;
```

Remarks：

初期化します。ACK タイプを引き渡します。

9.3 initWithData

Syntax：

```
- (instancetype)initWithData:(NSData *)data;
```

Remarks：

初期化します。バイナリデータからインスタンスを生成するメソッドです。

10. JvmaNAKData Class

typedef enum

```
typedef NS_ENUM(NSInteger, JvmaNAKStatus) {
    VmBusy              = 0x01,//JVMA モジュールがビジー状態であることを表す。
    VmStop              = 0x02,//自動販売機から NAK が返送されたことを表す。通信セッションは中止される。
    VmTimeOut           = 0x03,//自動販売機が無応答またはタイムアウトしたことを表す。通信セッションは中止される。
    VmError             = 0x04,//パケット形式または BCC エラーを検出したことを表す。通信セッションは中止される。
    JvmaTimeOut         = 0x05,//ホストと JVMA モジュール間の応答が確認できずタイムアウトしたことを表す。通信セッションは中断される。
    VmBeforeBuildSession = 0x06,//パスワードテキストによる通信セッション確立前に コマンドテキストが送信されたことを表す。
    JvmaDataSend        = 0x07,//コマンドテキストの前にホストからデータテキスト が送信されたことを表す。通信セッションは中断さ れる。
    JvmaOtherDataSend   = 0x08,//データテキストの中継転送中にホストから他のテキ ストが送信されたことを表す。通信セッションは中 断される。
    VmOutResponse       = 0x09,//ホストから応答待ち以外のタイミングで NAK を受信 した。通信セッションは中断される。
    SDKTimeOut          = 0x93,// SDKがAsReaderからの応答を受け取らなくなった場合、タイムアウトが発生したことを表す。
    VmOtherError        = 0x99 //その他のエラー。
};//NAK タイプ
```

10.1 Property

```
@property (nonatomic ,assign, readonly) textHeader nak;
//NAK ヘッダー
```

```
@property (nonatomic ,assign, readonly) JvmaNAKStatus status;
//NAK タイプ
```

```
@property (nonatomic, strong, readonly) NSData  *hexData;
//NAK データ
```

```
@property (nonatomic, strong, readonly) NSString *nakMessage;
//NAK エラーメッセージ
```

```
@property (nonatomic, strong ,readonly) NSString *hexStatus;
//当面ステータスの 16 進数文字列
```

10.2 initWithStatus

Syntax :

```
- (instancetype)initWithStatus:( JvmaNAKStatus)status;
```

Remarks :

初期化します。ACK タイプを引き渡します。

10.3 initWithData

Syntax :

```
- (instancetype)initWithData:(NSData *)data;
```

Remarks :

初期化します。バイナリデータからインスタンスを生成するメソッドです。