M3 MOBILE

WR15 SDK Integration Guide

V1.2

2025/11/21

Revision History

Date	Version	Comments	Author	Reviewer
2025/10/21	V1.0	Initial	Seongrak_Choi	
2025/11/13	V1.1	Changed SDK distribution	Seongrak_Choi	
2025/11/21	V1.2	Add Error Code	Seongrak_Choi	

Index

1. Import SDK

1.1 Libraries

1. Add the Maven repository URL to **settings.gradle.kts**

```
dependencyResolutionManagement {
    repositories {
        google()
        mavenCentral()

        //Add this line
        maven("https://m3mobile.github.io/wr15-sdk-maven")
    }
}
```

* If you are using **Groovy**

```
dependencyResolutionManagement {
    repositories {
        google()
        mavenCentral()

        // Add this line

        maven { url "https://m3mobile.github.io/wr15-sdk-maven" }
    }
}
```

2. Add the appropriate SDK version to **build.gradle (Module: app)**

```
dependencies {
    implementation("com.m3:wr15-sdk:1.2.0") // Update to the appropriate version
}
```

2. Permissions Required

Bluetooth and location permissions are required to use our SDK. Declare the appropriate permissions in your app according to the Android versions you target.

2.1 When Android 12+ (API 31+) is the primary target.

```
<uses-permission android:name="android.permission.BLUETOOTH_SCAN"
    android:usesPermissionFlags="neverForLocation"/>
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT"/>
```

2.2 When Android 6–11 (API 25–30) must also be supported.

```
<!-- Android 6~11 (API 25~30) -->
<uses-permission android:name="android.permission.BLUETOOTH"
   android:maxSdkVersion="30"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"
   android:maxSdkVersion="30"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
   android:maxSdkVersion="30"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
   android:maxSdkVersion="30"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
   android:maxSdkVersion="30"/>

<!-- Android 12+ (API 31+) -->
<uses-permission android:name="android.permission.BLUETOOTH_SCAN"
   android:usesPermissionFlags="neverForLocation"/>
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT"/>
```

2.3 Handling runtime permission requests is mandatory.

- BLUETOOTH_SCAN

- BLUETOOTH_CONNECT

- ACCESS_FINE_LOCATION

- ACCESS_COARSE_LOCATION

Because the four permissions above are runtime permissions that require explicit user consent, make sure to verify that the Bluetooth permissions have been granted before using the SDK.

3. Initialize SDK

3.1 Initialize the dependence libraries

Before using the library, make sure to call initialize to perform the required initialization.

Kotlin

```kotlin
val config = M3Wr15SdkConfig.builder()
            .connectionType(ConnectionType.SPP)
            .enableLog(true)
            .build()


M3Wr15Sdk.initialize(context, config)

```

Java

```java
M3Wr15SdkConfig config = M3Wr15SdkConfig.builder()
        .connectionType(ConnectionType.SPP)
        .enableLog(true)
        .build();

M3Wr15Sdk sdk =
        M3Wr15Sdk.initialize(context, config);
```

3.1.1 M3Wr15SdkConfig

- *connectionType(): Communication method with the WR15*

*** ConnectionType.BLE is not supported in v1.0**

- *enableLog(): Whether the SDK outputs logs.*

3.1.2 Context

Because the context is used for scanning and SharedPreferences, pass an **Application context rather than an Activity context.**

4. Search for WR15

4.1 S2P (Scan to Pair)

Scan the S2P barcode with the WR15 to discover the corresponding device.

4.1.1 Generate Barcode

```
/* M3Wr15Sdk */

@JvmOverloads
@JvmStatic
fun getBarcodeBitmapForS2P(width: Int = 260, height: Int = 50): Bitmap?

@JvmStatic
fun getCmdForS2P(): String
```

In typical cases, you can generate the S2P barcode using the bitmap returned by ***getBarcodeBitmapForS2P( )***. If you need a different barcode type, use the String returned by ***getCmdForS2P( )*** to create the barcode in your desired format.

4.1.2 Start S2P Scan

An S2P scan must be in progress when the WR15 ring scanner scans the barcode generated above.

For Kotlin

```
/* M3Wr15Sdk */

@RequiresPermission(
allOf = [Manifest.permission.BLUETOOTH_SCAN, Manifest.permission.BLUETOOTH_CONNECT])
fun startS2PScan(
    onDeviceFound: (BluetoothDevice) -> Unit,
    onScanFailed: (errorCode: Int) -> Unit
)
```

For Java

```
/* M3Wr15Sdk */

@JvmStatic
@RequiresPermission(
allOf = [Manifest.permission.BLUETOOTH_SCAN, Manifest.permission.BLUETOOTH_CONNECT])
```

```
fun startS2PScan(listener: S2PScanResultListener)

interface S2PScanResultListener {
    fun onDeviceFound(device: BluetoothDevice)
    fun onScanFailed(errorCode: Int)
}
```

The ***startS2PScan( )*** method requires **BLUETOOTH_SCAN** and **BLUETOOTH_CONNECT** to be granted; add a guard to verify permissions are granted before invoking it.

4.1.3 Stop S2P Scan

Stop ***startS2PScan()*** immediately after the WR15 scanner is discovered. It is also recommended to stop ***startS2PScan()*** when navigating away from the screen that displays the S2P barcode.

```
/* M3Wr15Sdk */

@RequiresPermission(Manifest.permission.BLUETOOTH_SCAN)
fun stopS2PScan()
```

Example

```
@RequiresPermission(Manifest.permission.BLUETOOTH_SCAN)
fun stopS2PScan()

M3Wr15Sdk.startS2PScan(object: S2PScanResultListener) {
    override fun onDeviceFound(device: BluetoothDevice) {
        M3Wr15Sdk.stopS2PScan() // Stop scanning
       // Attempt to connect to the discovered BluetoothDevice.

    }

    override fun onScanFailed(errorCode: Int) {
        // Handle scan failures
    }
})
```

4.2 Search and Connect

Search for nearby WR15 scanner devices.

4.2.1 Start Discovery Scan

Because it runs as a discovery scan, the scan automatically stops approximately 20 seconds after it starts.

```
/*M3Wr15Sdk*/

@RequiresPermission(allOf = [Manifest.permission.BLUETOOTH_SCAN,
Manifest.permission.BLUETOOTH_CONNECT])
```

```
fun startDiscoveryScan(listener: DiscoveryScanResultListener)

interface DiscoveryScanResultListener {
    fun onStarted() //Start scanning
    fun onError() // An error occurred
    Fun onDeviceFound(device: DiscoveryDeviceModel) //WR15 scanner device discovered
    fun onFinished() //Stop scanning
}
```

Exmaple

```
M3Wr15Sdk.startDiscoveryScan(object : DiscoveryScanResultListener {
    @RequiresPermission(Manifest.permission.BLUETOOTH_CONNECT)
    override fun onDeviceFound(device: DiscoveryDeviceDodel) {
   // Handle device discovery.

    }

    override fun onError(errorMsg: String) {
    // Handle Error
    }

    override fun onFinished() {
    // Handle finish scanning
    }

    override fun onStarted() {
    // Handle start scanning
    }
}
)
```

4.2.2 Stop Discovery Scan

Stop the discovery scan.

```
/*M3Wr15Sdk*/

@JvmStatic
fun stopDiscoveryScan()
```

5. Connect to WR15

5.1 Connect

Connect the BluetoothDevice to the SDK.

For Kotlin

```kotlin
/*M3Wr15Sdk*/

@RequiresPermission(Manifest.permission.BLUETOOTH_CONNECT)
fun connectToDevice(
    device: BluetoothDevice,
    onSuccess: (session: TransportSession) -> Unit,
    onFailure: (errorCode: Int, errorMsg: String) -> Unit
)
```

For Java

```kotlin
/*M3Wr15Sdk*/

@JvmStatic
@RequiresPermission(Manifest.permission.BLUETOOTH_CONNECT)
fun connectToDevice(device: BluetoothDevice, listener: ConnectResultListener)

interface ConnectResultListener {
    fun onSuccess(session: TransportSession)
    fun onFailure(errorCode: Int, errorMsg: String)
}
```

You can use the TransportSession returned by ***onSuccess(session: TransportSession)*** to communicate with the WR15 ring scanner later.

Example

```kotlin
val sdk = M3Wr15Sdk.initialize(applicationContext, config)

sdk.startS2PScan(object: DeviceFindResultListener {
    override fun onDeviceFound(device: BluetoothDevice) {
        sdk.stopS2PScan()
        sdk.connectToDevice(device, object:ConnectResultListener {
            override fun onSuccess(session: TransportSession) {
                //Handle Success
            }
            override fun onFailure(errorMsg: String) {
                //Handle Failure
            }
        })
    }

    override fun onScanFailed(errorCode: Int) {
        // Handle Scan failures
    }
```

```
19. })
```

6. Disconnect

6.1 Disconnect to WR15

Disconnect from the currently connected WR15 scanner.

```
1. /* TransportSession */
2.
3. //For Kotlin
4. fun disconnect(onSuccess: () -> Unit, onFailure: (errorMsg: String) -> Unit)
5.
6. //For Java
7. fun disconnect(listener: DisconnectResultListener)
```

6.2 Observe Disconnect

Detect when the connection is disconnected.

```
1. /* TransportSession */
2.
3. fun addDisconnectListener(disconnectListener: DisconnectListener)
4. fun removeDisconnectListener(disconnectListener: DisconnectListener)
5.
6. fun interface DisconnectListener {
7.     fun onDisconnect()
8. }
```

Example

```
1. val disconnectListener = object: DisconnectListener {
2.     override fun onDisconnect() {
3.     // Handle disconnection
4.     }
5.   }
6. fun connectToDevice() {
7.     sdk.connectToDevice(device, object:ConnectResultListener {
8.         override fun onSuccess(session:TransportSession) {
9.             session.addDisconnectListener(disconnectListener)
10.        }
11.        override fun onFailure(errorMsg: String) {
12.            //Handle Failure
13.        }
14.    })
15. }
16.
```

```
fun disconnect() {
    if (sdk.isTransportSessionInitialized()) { // Attempt to disconnect only if the session has been initialized
        val session = sdk.getTransportSession()
        session .disconnect(
            onSuccess = {
                // Handle successful disconnection
                session.removeDisconnectListener(disconnectListener)
            },
            onFailure = { errorMsg ->
                // Handle disconnection failures
            }
        )
    }
}
```

To prevent memory leaks, make sure to unregister registered listeners using ***removeDisconnectListener( )*** when the connection is terminated or when the listener is no longer needed.

* You must pass the same listener instance registered with addDisconnectListener as the parameter to removeDisconnectListener.

7. Receive Barcode Data

7.1 Add Barcode Data

To receive decoded barcode data from the WR15 ring scanner, you must register a listener.

```
/* TransportSession */

fun addDecodingDataListener(decodingDataListener: DecodingDataListener)

interface DecodingDataListener {
    fun onReceive(decodingData: ByteArray)
}
```

7.2 Remove Barcode Data

Unregister the BarcodeDataListener.

```
/* TransportSession */

fun removeBarcodeDataListener(barCodeDataListener: BarcodeDataListener)
```

To prevent memory leaks, always unregister the listener when the connection is terminated or when the listener is no longer needed.

* You must pass the same listener instance registered with addBarcodeDataListener as the parameter to removeBarcodeDataListener.

8. Scanner Device Information

Device and scanner configuration information for the WR15 ring scanner is provided as the ***DeviceState*** **data class**.

For details on ***DeviceState***, see section 9, DeviceState

8.1 Observe DeviceState

DeviceState is delivered via a callback and is re-sent whenever settings change.

```
/* TransportSession */

fun addDeviceStateListener(listener: DeviceStateListener)
fun removeDeviceStateListener(listener: DeviceStateListener)

interface DeviceStateListener {
    fun onDeviceStateChanged(state: DeviceState)
}

data class DeviceState(
    val deviceInfo: DeviceInfo = DeviceInfo(),
    val general: GeneralSettings = GeneralSettings(),
    val symbologies: Symbologies = Symbologies(),
    val basicFormat: BasicFormat = BasicFormat(),
    val readerSettings: ReaderSettings = ReaderSettings(),
    val devSettings: DevSettings = DevSettings()
)
```

8.2 Set Scanner Setting

Modify the WR15 scanner settings

For Kotlin

```

fun setSetting(
    setting: InternalSettingCommand,
    onSuccess: () -> Unit,
    onFailure: (errorCode: Int, errorMsg: String) -> Unit,
)
```

For Java

```
/* TransportSession */

fun setSetting(setting: SettingCommand, listener: SetSettingResultListener)
```

```
4.
5.  interface SetSettingResultListener{
6.      fun onSuccess()
7.      fun onFailure(errorCode: Int, errorMsg: String)
8.  }
```

8.2.1 SettingCommand

SettingCommand is used as the parameter to ***setSetting()*** and encapsulates the target setting and its options. Create a ***SettingCommand*** using the Settings object.

8.2.2 Generate SettingCommand

Settings is the API entry point for easily creating ***SettingCommand*** instances.

```
1.  object Settings {
2.      object Codabar {
3.          @JvmStatic
4.          fun setEnable(enable: Boolean): SettingCommand
5.          @JvmStatic
6.          fun setLength1(length: Int): SettingCommand
7.          ...
8.      }
9.
10.     object Code11 {
11.         @JvmStatic
12.         fun setEnable(enable: Boolean): SettingCommand
13.         @JvmStatic
14.         fun setLength1(length: Int): SettingCommand
15.         ...
16.      }
17.
18.      object General {
19.         @JvmStatic
20.         fun findMe(): SettingCommand
21.         @JvmStatic
22.         fun setAimer(enable: Boolean): SettingCommand
23.         @JvmStatic
24.         fun setIllumination(enable: Boolean): SettingCommand
25.         @JvmStatic
26.         fun setSound(type: SoundType): SettingCommand
27.         @JvmStatic
28.         fun setVibrate(enable: Boolean): SettingCommand
29.         ...
30.       }
31.      ... //Other settings...
32. }
```

For more detailed configuration items, see section 10.Settings.

Example

For Kotlin

```kotlin
session.setSetting(
    Settings.EAN13.setEnable(true),
    onSuccess = {
      // Handle Successful set
    },
    onFailure = { errorMsg ->
      // Handle setting failures
    }
)
```

For Java

```java
session.setSetting(
    Settings.General.setSound(SoundType.OFF),
    object : SetSettingResultListener {
        override fun onFailure(errorMsg: String) {
          // Handle setting failures
        }

        override fun onSuccess() {
          // Handle Successful set
        }
    }
)
```

8.2.3 Caution

Configurable settings and option values vary by WR15 ring scanner model (e.g, E4770, SE4107). Review the documentation carefully and request only the settings applicable to the selected model.

If a setting not supported by the currently used WR15 ring scanner is requested, the ***setSetting*** callback will be invoked with ***onFailure( )***.

9. DeviceState

DeviceState is an object that contains all scanner information. For options not supported by the WR15 scanner in use, the value will be null or an empty string.

**Type definitions can be found in section 10. Settings.*

9.1 Strucutre

```
data class DeviceState(
    val deviceInfo: DeviceInfo = DeviceInfo(),
    val general: GeneralSettings = GeneralSettings(),
    val symbologies: Symbologies = Symbologies(),
    val basicFormat: BasicFormat = BasicFormat(),
    val readerSettings: ReaderSettings = ReaderSettings(),
    val devSettings: DevSettings = DevSettings()
)
```

9.1.1 DeviceInfo

```
data class DeviceInfo(
    val name: String? = null,
    val macAddress: String? = null,
    val model: String? = null,
    val serialNumber: String? = null,
    val fwVersion: String? = null,
    val releaseDate: String? = null,
    val batteryPercent: Int? = null,
    val scannerId: ScannerIdType? = null,
    val btMode: BtModeType? = null,
)
```

9.1.2 GeneralSettings

```
data class GeneralSettings(
    val sound: SoundType? = null,
    val vibrateEnabled: Boolean? = null,
    val ledEnabled: Boolean? = null,
    val buttonEnabled: Boolean? = null,
    val aimEnabled: Boolean? = null,
    val illuminationEnabled: Boolean? = null,
)
```

9.1.3 Symbologies

```
data class Symbologies(
    val australianPostalVals: AustralianVals = AustralianVals(),
    val aztecVals: AztecVals = AztecVals(),
    val chinese2Of5Vals: Chinese2Of5Vals = Chinese2Of5Vals(),
    val codabarVals: CodabarVals = CodabarVals(),
```

```
    val code11Vals: Code11Vals = Code11Vals(),
    val code39Vals: Code39Vals = Code39Vals(),
    val code93Vals: Code93Vals = Code93Vals(),
    val code128Vals: Code128Vals = Code128Vals(),
    val compositeVals: CompositeVals = CompositeVals(),
    val dataMatrixVals: DataMatrixVals = DataMatrixVals(),
    val discrete2Of5Vals: Discrete2Of5Vals = Discrete2Of5Vals(),
    val dotCodeVals: DotCodeVals = DotCodeVals(),
    val ean8Vals: Ean8Vals = Ean8Vals(),
    val ean13Vals: Ean13Vals = Ean13Vals(),
    val gs1DataBar14Vals: Gs1DataBar14Vals = Gs1DataBar14Vals(),
    val gs1DataBarExpandedVals: Gs1DataBarExpandedVals = Gs1DataBarExpandedVals(),
    val gs1DataBarLimitedVals: Gs1DataBarLimitedVals = Gs1DataBarLimitedVals(),
    val gs1128Vals: Gs1128Vals = Gs1128Vals(),
    val hanXinVals: HanXinVals = HanXinVals(),
    val interleaved2Of5Vals: Interleaved2Of5Vals = Interleaved2Of5Vals(),
    val isbt128Vals: Isbt128Vals = Isbt128Vals(),
    val japanesePostalVals: JapanesePostalVals = JapanesePostalVals(),
    val korean3Of5Vals: Korean3Of5Vals = Korean3Of5Vals(),
    val matrix2Of5Vals: Matrix2Of5Vals = Matrix2Of5Vals(),
    val maxiCodeVals: MaxiCodeVals = MaxiCodeVals(),
    val microPdf417Vals: MicroPdf417Vals = MicroPdf417Vals(),
    val microQrCodeVals: MicroQrCodeVals = MicroQrCodeVals(),
    val msiVals: MsiVals = MsiVals(),
    val netherlandsKixVals: NetherlandsKixVals = NetherlandsKixVals(),
    val pdf417Vals: Pdf417Vals = Pdf417Vals(),
    val qrCodeVals: QrCodeVals = QrCodeVals(),
    val ukPostalVals: UkPostalVals = UkPostalVals(),
    val upcAVals: UpcAVals = UpcAVals(),
    val upcE1Vals: UpcE1Vals = UpcE1Vals(),
    val upcEanVals: UpcEanVals = UpcEanVals(),
    val upcEVals: UpcEVals = UpcEVals(),
    val upuFicsPostalVals: UpuFicsPostalVals = UpuFicsPostalVals(),
    val usPlanetVals: UsPlanetVals = UsPlanetVals(),
    val usPostnetVals: UsPostnetVals = UsPostnetVals(),
)
```

AustralianVals

Supported Scanners : ***SE4107, SE5500***

```
data class AustralianVals(
    val enabled: Boolean? = null,
)
```

AztecVals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class AztecVals(
    val enabled: Boolean? = null
)
```

Chinese2Of5Vals

Supported Scanners : ***SE4107, SE5500***

```
data class Chinese2Of5Vals(
    val enabled: Boolean? = null
)
```

CodabarVals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class CodabarVals(
    val enabled: Boolean? = null,
    val length1: Int? = null,
    val length2: Int? = null,
    val clsiEditing: Boolean? = null, //E4770 is not supported
    val notisEditing: Boolean? = null,
    val securityLevel: CodabarType.SecurityType? = null, //E4770 is not supported
)
```

Code11Vals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class Code11Vals(
    val enabled: Boolean? = null,
    val length1: Int? = null,
    val length2: Int? = null,
    val reportCheckDigit: Boolean? = null,
    val verifyCheckDigit: Code11Type.VerifyCheckDigitType? = null
)
```

Code39Vals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class Code39Vals(
    val enabled: Boolean? = null,
    val triopticCode39: Boolean? = null, //E4770 is not supported
    val length1: Int? = null,
    val length2: Int? = null,
    val reducedQuietZone: Boolean? = null, //E4770 is not supported
    val convertToCode32: Boolean? = null,
    val fullAscii: Boolean? = null,
    val reportCheckDigit: Boolean? = null, // true = TRANSMIT
    val reportCode32Prefix: Boolean? = null,
    val securityLevel: Code39Type.SecurityType? = null, //E4770 is not supported
    val verifyCheckDigit: Boolean? = null  // true = ENABLE
)
```

Code93Vals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class Code93Vals(
    val enabled: Boolean? = null,
    val length1: Int? = null,
    val length2: Int? = null
)
```

Code128Vals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class Code128Vals(
    val enabled: Boolean? = null,
    val length1: Int? = null,
    val length2: Int? = null,
    val reducedQuietZone: Boolean? = null, //E4770 is not supported
    val ignoreFnc4: Boolean? = null, //E4770 is not supported
    val checkIsbtTable: Boolean? = null, //E4770 is not supported
    val isbt128ConcatMode: Code128Type.ISBT128ConcatModeType? = null, //E4770 is not supported
    val securityLevel: Code128Type.SecurityType? = null, //E4770 is not supported
    val emulationMode: Boolean? = null//E4770 is not supported
)
```

CompositeVals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class CompositeVals(
    val enabled: Boolean? = null, //SE4107, SE5500 is not supported
    val compositeABEnabled: Boolean? = null, //E4770 is not supported
    val compositeCEnabled: Boolean? = null, //E4770 is not supported
    val tlc39Enable: Boolean? = null, //E4770 is not supported
    val upcCompositeMode: CompositeType.UPCCompositeModeType? = null //E4770 is not supported
)
```

On the E4770, the single Enabled setting consolidates **ABEnabled** and **CEnabled**. On the SE4107 and SE5500, setting both **ABEnabled** and **CEnabled** to true is equivalent to ***enabled = true***.

DataMatrixVals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class DataMatrixVals(
    val enabled: Boolean? = null,
    val inverse: DataMatrixType.InverseType? = null,
    val decodeMirrorImages: DataMatrixType.DecodeMirrorImagesType? = null, //E4770 is not supported
    val gs1DataMatrix: Boolean? = null //E4770 is not supported
)
```

Discrete2Of5Vals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class Discrete2Of5Vals(
    val enabled: Boolean? = null,
    val length1: Int? = null,
    val length2: Int? = null
)
```

DotCodeVals

Supported Scanners : ***SE4107, SE5500***

```
data class DotCodeVals(
    val enabled: Boolean? = null,
    val inverse: DotCodeType.InverseType? = null,
    val mirror: DotCodeType.MirrorType? = null,
    val prioritize: Boolean? = null
)
```

Ean8Vals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class Ean8Vals(
    val enabled: Boolean? = null
)
```

Ean13Vals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class Ean13Vals(
    val enabled: Boolean? = null
)
```

Gs1DataBar14Vals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class Gs1DataBar14Vals(
    val enabled: Boolean? = null
)
```

Gs1DataBarExpandedVals

Supported Scanners : ***SE4107, SE5500***

```
data class Gs1DataBarExpandedVals(
    val enabled: Boolean? = null
)
```

Gs1DataBarLimitedVals

Supported Scanners : ***SE4107, SE5500***

```
data class Gs1DataBarLimitedVals(
    val enabled: Boolean? = null,
    val securityLevel: Gs1DataBarLimitedType.SecurityType? = null
)
```

Gs1128Vals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class Gs1128Vals(
    val enabled: Boolean? = null
)
```

HanXinVals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class HanXinVals(
    val enabled: Boolean? = null,
    val inverse: HanXinType.InverseType? = null
)
```

Interleaved2Of5Vals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class Interleaved2Of5Vals(
    val enabled: Boolean? = null,
    val length1: Int? = null,
    val length2: Int? = null,
    val checkDigit: Interleaved2Of5Type.CheckDigitType? = null,
    val reportCheckDigit: Boolean? = null,     // true = TRANSMIT
    val securityLevel: Interleaved2Of5Type.SecurityType? = null, //E4770 is not supported
    val convertItf14ToEan13: Boolean? = null, //E4770 is not supported
    val reducedQuietZone: Boolean? = null //E4770 is not supported
)
```

Isbt128Vals

Supported Scanners : ***SE4107, SE5500***

```
data class Isbt128Vals(
    val enabled: Boolean? = null
)
```

JapanesePostalVals

Supported Scanners : ***SE4107, SE5500***

```
data class JapanesePostalVals(
    val enabled: Boolean? = null
)
```

Korean3Of5Vals

Supported Scanners : ***SE4107, SE5500***

```
data class Korean3Of5Vals(
    val enabled: Boolean? = null
)
```

Matrix2Of5Vals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class Matrix2Of5Vals(
    val enabled: Boolean? = null,
    val length1: Int? = null,
    val length2: Int? = null,
    val redundancy: Boolean? = null, //E4770 is not supported
    val reportCheckDigit: Boolean? = null,  // true = TRANSMIT
    val verifyCheckDigit: Boolean? = null   // true = ENABLE
)
```

MaxiCode

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class MaxiCodeVals(
    val enabled: Boolean? = null
)
```

MicroPdf417Vals

Supported Scanners : ***SE4107, SE5500***

```
data class MicroPdf417Vals(
    val enabled: Boolean? = null
)
```

MicroQrCodeVals

Supported Scanners : ***SE4107, SE5500***

```
data class MicroQrCodeVals(
    val enabled: Boolean? = null
)
```

MsiVals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class MsiVals(
    val enabled: Boolean? = null,
    val length1: Int? = null,
    val length2: Int? = null,
    val checkDigit: MsiType.CheckDigitType? = null,
    val checkDigitScheme: MsiType.CheckDigitSchemeType? = null,
    val reportCheckDigit: Boolean? = null                  // true = TRANSMIT
)
```

NetherlandKixVals

Supported Scanners : ***SE4107, SE5500***

```
data class NetherlandsKixVals(
    val enabled: Boolean? = null
)
```

Pdf417Vals

Supported Scanners : ***SE4107, SE5500***

```
data class Pdf417Vals(
    val enabled: Boolean? = null
)
```

QrCodeVals

Supported Scanners **: *SE4107, SE5500, E4770***

```
data class QrCodeVals(
    val enabled: Boolean? = null
)
```

UkPostalVals

Supported Scanners : ***SE4107, SE5500***

```
data class UkPostalVals(
    val enabled: Boolean? = null,
    val reportCheckDigit: Boolean? = null // true = TRANSMIT
)
```

UpcAVals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class UpcAVals(
    val enabled: Boolean? = null,
    val reportCheckDigit: Boolean? = null,     // true = TRANSMIT
    val preamble: UpcAType.PreambleType? = null,
    val upcA2BitsSupplementals: Boolean? = null,
```

```
    val upcA5BitsSupplementals: Boolean? = null
)
```

UpcE1Vals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class UpcE1Vals(
    val enabled: Boolean? = null,
    val convertToUpcA: Boolean? = null,// true = CONVERT //E4770 is not supported
    val preamble: UpcE1Type.PreambleType? = null, //E4770 is not supported
    val reportCheckDigit: Boolean? = null // true = TRANSMIT //E4770 is not supported
)
```

UPpcEanVals

Supported Scanners : ***SE4107, SE5500, E4770***

```
data class UpcEanVals(
    val reportEan8CheckDigit: Boolean? = null, // true = TRANSMIT
    val reportEan13CheckDigit: Boolean? = null, // true = TRANSMIT

    val supplementalMode: UpcEanType.SupplementalModeType? = null, //E4770 is not supported
    val supplementalAimIdFormat: UpcEanType.SupplementalAimIdFormatType? = null, //E4770 is not supported
    val reducedQuietZone: Boolean? = null, //E4770 is not supported
    val bookland: UpcEanType.BooklandType? = null, //E4770 is not supported
    val uccCouponExtend: Boolean? = null, //E4770 is not supported
    val couponReport: UpcEanType.CouponReportType? = null, //E4770 is not supported
    val issnEan: Boolean? = null, //E4770 is not supported
    val translateUpcAToEan13: Boolean? = null, //E4770 is not supported
    val supplementalRedundancy: Int? = null, //E4770 is not supported
    val supplemental2: Boolean? = null, //E4770 is not supported
    val supplemental5: Boolean? = null, //E4770 is not supported

    val ean8ReadSupplements: Boolean? = null, //SE4107, SE5500 is not supported
    val ean13ReadSupplements: Boolean? = null, // //SE4107, SE5500 is not supported
    val ean8Supplemental2: Boolean? = null, // //SE4107, SE5500 is not supported
    val ean13Supplemental2: Boolean? = null, // //SE4107, SE5500 is not supported
    val ean8Supplemental5: Boolean? = null, // //SE4107, SE5500 is not supported
    val ean13Supplemental5: Boolean? = null // //SE4107, SE5500 is not supported
)
```

On the SE4107 and SE5500, the **Supplemental2** feature is split on the E4770 into **ean8Supplemental2** and **ean8Supplemental5**, and the **Supplemental5** feature is split into **ean13Supplemental2** and **ean13Supplemental5**.

Additionally, to use **ean8Supplemental2** and **ean8Supplemental5** on the E4770, you must set **ean8ReadSupplements** to true. Likewise, to use **ean13Supplemental2** and **ean13Supplemental5**, you must set **ean13ReadSupplements** to true.

UpcEVals

지원하는 Scanner : ***SE4107, SE5500, E4770***

```
data class UpcEVals(
    val enabled: Boolean? = null,
```

```
    val convertToUpcA: Boolean? = null,   // true = CONVERT
    val preamble: UpcEType.PreambleType? = null,
    val reportCheckDigit: Boolean? = null, // true = TRANSMIT
    val upcE2BitsSupplementals: Boolean? = null,
    val upcE5BitsSupplementals: Boolean? = null
)
```

UpuFicsPostalVals

Supported Scanners : ***SE4107, SE5500***

```
data class UpuFicsPostalVals(
    val enabled: Boolean? = null
)
```

UsPlanetVals

Supported Scanners : ***SE4107, SE5500***

```
data class UsPlanetVals(
    val enabled: Boolean? = null,
    val reportCheckDigit: Boolean? = null // true = TRANSMIT
)
```

UsPostnetVals

Supported Scanners : ***SE4107, SE5500***

```
data class UsPostnetVals(
    val enabled: Boolean? = null
)
```

9.1.4 BasicFormat

```
data class BasicFormat(
    val transmitCodeId: BasicFormatType.TransmitCodeIdType? = null,
    val endChar: BasicFormatType.EndCharType? = null,
    val substringFormation: String? = null,
    val removeFnc: Boolean? = null,
    val translateData: String? = null,
    val prefixPostfixAsAsciiHex: Boolean? = null,
    val prefix: String? = null,
    val postfix: String? = null
)
```

9.1.5 readerSettings

```
data class ReaderSettings(
    val readMode: ReaderType.ReadModeType? = null,
    val laserOnTime: Int? = null,
    val inverse1D: ReaderType.Inverse1DType? = null,
    val quietZoneLevelFor1D: ReaderType.QuietZoneLevelType? = null,
    val poorQualityDecodeEffort: ReaderType.PoorQualityDecodeEffortType? = null,
)
```

9.1.6 DevSettings

```
data class DevSettings(
    val batteryCallback: Boolean? = null
)
```

10. Settings

Settings is the SDK entry point for easily creating ***SettingCommand*** instances for scanner configuration. Per-symbology options and common reader options are wrapped with a consistent set of methods, allowing you to construct safe commands with a simple call.

10.1 AustralianPostal

```
object AustralianPostal {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.2 Aztec

```
object Aztec {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.3 Chinese2of5

```
object Chinese2of5 {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.4 Codabar

```
object Codabar {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setLength1(length: Int): SettingCommand

    @JvmStatic
    fun setLength2(length: Int): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setCLSIEditing(enable: Boolean): SettingCommand

    @JvmStatic
    fun setNotisEditing(enable: Boolean): SettingCommand

```

```
    @JvmStatic //E4770 is not supported
    fun setSecurityLevel(value: CodabarType.SecurityType): SettingCommand
}
```

10.4.1 CodabarType

```
object CodabarType {
    enum class SecurityType : TypeValue {
        LEVEL_0, LEVEL_1, LEVEL_2, LEVEL_3
    }
}
```

10.5 Code11

```
object Code11 {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setLength1(length: Int): SettingCommand

    @JvmStatic
    fun setLength2(length: Int): SettingCommand

    @JvmStatic
    fun setReportCheckDigit(enable: Boolean): SettingCommand

    @JvmStatic
    fun setVerifyCheckDigit(value: Code11Type.VerifyCheckDigitType): SettingCommand
}
```

10.5.1 Code11Type

```
object Code11Type {
    enum class VerifyCheckDigitType : TypeValue {
        DISABLE,
        ONE_CHECK_DIGIT,
        TWO_CHECK_DIGITS
    }
}
```

10.6 Code39

```
object Code39 {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setTriopticCode39(enable: Boolean): SettingCommand

    @JvmStatic
```

```
    fun setLength1(length: Int): SettingCommand

    @JvmStatic
    fun setLength2(length: Int): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setReducedQuietZone(enable: Boolean): SettingCommand

    @JvmStatic
    fun setConvertToCode32(enable: Boolean): SettingCommand

    @JvmStatic
    fun setFullAscii(enable: Boolean): SettingCommand

    @JvmStatic
    fun setReportCheckDigit(enable: Boolean): SettingCommand

    @JvmStatic
    fun setReportCode32Prefix(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setSecurityLevel(value: Code39Type.SecurityType): SettingCommand

    @JvmStatic
    fun setVerifyCheckDigit(enable: Boolean): SettingCommand
}
```

10.6.1 Code39Type

```
object Code39Type {
    enum class SecurityType : TypeValue {
        LEVEL_0, LEVEL_1, LEVEL_2, LEVEL_3
    }
}
```

10.7 Code93

```
object Code93 {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setLength1(length: Int): SettingCommand

    @JvmStatic
    fun setLength2(length: Int): SettingCommand
}
```

10.8 Code128

```
object Code128 {
    @JvmStatic
```

```
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setLength1(length: Int): SettingCommand

    @JvmStatic
    fun setLength2(length: Int): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setReducedQuietZone(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setIgnoreFnc4(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setCheckIsbtTable(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setIsbt128ConcatMode(value: Code128Type.ISBT128ConcatModeType): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setSecurityLevel(value: Code128Type.SecurityType): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setEmulationMode(enable: Boolean): SettingCommand
}
```

10.8.1 Code128Type

```
object Code128Type {
    enum class SecurityType : TypeValue {
        LEVEL_0, LEVEL_1, LEVEL_2, LEVEL_3
    }

    enum class ISBT128ConcatModeType : TypeValue {
        ENABLE, DISABLE, AUTO
    }
}
```

10.9 Composite

```
object Composite {
    @JvmStatic //SE4107, SE5500 is not supported
    fun setEnable(enable: Boolean): SettingCommand //SE4107, SE5500 is not supported

    @JvmStatic //E4770 is not supported
    fun setABEnable(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setCEnable(value: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setTlc39Enable(enable: Boolean): SettingCommand

```

```
    @JvmStatic //E4770 is not supported
    fun setUpcCompositeMode(value: CompositeType.UPCCompositeModeType): SettingCommand
}
```

- ***setEnable(enable: Boolean)*** applies both ***setABEnable(enable)*** and ***setCEnable(enable)***. For the E4770 scanner, ***setEnable(true)*** enables both AB and C, and ***setEnable(false)*** disables both.

10.9.1 CompositeType

```
object CompositeType {
    enum class UPCCompositeModeType : TypeValue {
        NEVER_LINKED, ALWAYS_LINKED, AUTO_DISCRIMINATE
    }
}
```

10.10 DataMatrix

```
object DataMatrix {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setInverse(value: DataMatrixType.InverseType): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setDecodeMirrorImages(value: DataMatrixType.DecodeMirrorImagesType): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setGs1DataMatrix(enable: Boolean): SettingCommand
}
```

10.10.1 DatatMatrixType

```
object DataMatrixType {
    enum class InverseType : TypeValue {
        REGULAR_ONLY, INVERSE_ONLY, AUTO
    }

    enum class DecodeMirrorImagesType : TypeValue {
        NEVER, ALWAYS, AUTO
    }
}
```

10.11 Discrete2Of5

```
object Discrete2Of5 {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setLength1(length: Int): SettingCommand
```

```
7.
8.      @JvmStatic
9.      fun setLength2(length: Int): SettingCommand
10. }
```

10.12 DotCode

```
1.  object DotCode {
2.      @JvmStatic //E4770 is not supported
3.      fun setEnable(enable: Boolean): SettingCommand
4.
5.      @JvmStatic //E4770 is not supported
6.      fun setInverse(value: DotCodeType.InverseType): SettingCommand
7.
8.      @JvmStatic //E4770 is not supported
9.      fun setMirror(value: DotCodeType.MirrorType): SettingCommand
10.
11.     @JvmStatic //E4770 is not supported
12.     fun setPrioritize(enable: Boolean): SettingCommand
13. }
```

10.12.1 DotCodeType

```
1.  object DotCodeType {
2.      enum class InverseType : TypeValue { ENABLE, DISABLE, AUTODETECT }
3.      enum class MirrorType : TypeValue { REGULAR, INVERSE_ONLY, AUTODETECT }
4.  }
```

10.13 EAN8

```
1.  object EAN8 {
2.      @JvmStatic
3.      fun setEnable(enable: Boolean): SettingCommand
4.  }
```

10.14 EAN13

```
1.  object EAN13 {
2.      @JvmStatic
3.      fun setEnable(enable: Boolean): SettingCommand
4.  }
```

10.15 Gs1DataBar14

```
object Gs1DataBar14 {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.16 Gs1DataBarExpanded

```
object Gs1DataBarExpanded {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.17 Gs1DataBarLimited

```
object Gs1DataBarLimited {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setLimitedSecurityLevel(value: Gs1DataBarLimitedType.SecurityType): SettingCommand
}
```

10.17.1 Gs1DataBarLimitedType

```
object Gs1DataBarLimitedType {
    enum class SecurityType : TypeValue {
        LEVEL_0, LEVEL_1, LEVEL_2, LEVEL_3
    }
}
```

10.18 Gs1128

```
object Gs1128 {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.19 HanXin

```
object HanXin {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

```

```
    @JvmStatic
    fun setInverse(value: HanXinType.InverseType): SettingCommand
}
```

10.20 Interleaved2Of5

```
object Interleaved2Of5 {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setLength1(length: Int): SettingCommand

    @JvmStatic
    fun setLength2(length: Int): SettingCommand

    @JvmStatic
    fun setCheckDigit(value: Interleaved2Of5Type.CheckDigitType): SettingCommand

    @JvmStatic
    fun setReportCheckDigit(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setSecurityLevel(value: Interleaved2Of5Type.SecurityType): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setConvertItf14ToEan13(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setReducedQuietZone(enable: Boolean): SettingCommand
}
```

10.20.1 Interleaved2Of5Type

```
object Interleaved2Of5Type {
    enum class CheckDigitType : TypeValue {
        DISABLE,
        ENABLE,
        USS_CHECK_DIGIT, //E4770 is not supported
        OPCC_CHECK_DIGIT //E4770 is not supported
    }

    enum class SecurityType : TypeValue {
        LEVEL_0, LEVEL_1, LEVEL_2, LEVEL_3
    }
}
```

10.21 ISBT128

```
object ISBT128 {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand
```

```
}
```

10.22 JapanesePostal

```
object JapanesePostal {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.23 Korean3of5

```
object Korean3of5 {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.24 Matrix2of5

```
object Matrix2Of5 {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setLength1(length: Int): SettingCommand

    @JvmStatic
    fun setLength2(length: Int): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setRedundancy(enable: Boolean): SettingCommand

    @JvmStatic
    fun setReportCheckDigit(enable: Boolean): SettingCommand

    @JvmStatic
    fun setVerifyCheckDigit(enable: Boolean): SettingCommand
}
```

10.25 MaxiCode

```
object MaxiCode {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.26 MicroPdf417

```
object MicroPdf417 {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.27 MicroQrCode

```
object MicroQrCode {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.28 MSI

```
object MSI {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setLength1(length: Int): SettingCommand

    @JvmStatic
    fun setLength2(length: Int): SettingCommand

    @JvmStatic
    fun setCheckDigit(value: MsiType.CheckDigitType): SettingCommand

    @JvmStatic
    fun setCheckDigitScheme(value: MsiType.CheckDigitSchemeType): SettingCommand

    @JvmStatic
    fun setReportCheckDigit(enable: Boolean): SettingCommand
}
```

10.28.1 MsiType

```
object MsiType {
    enum class CheckDigitType : TypeValue {
        ONE_CHECK_DIGIT, TWO_CHECK_DIGITS
    }

    enum class CheckDigitSchemeType : TypeValue {
        MOD_10_10, MOD_10_11
    }
}
```

10.29 NetherlandsKix

```
object NetherlandsKix {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.30 Pdf417

```
object Pdf417 {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.31 QrCode

```
object QrCode {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.32 UKPostal

```
object UKPostal {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setReportCheckDigit(enable: Boolean): SettingCommand
}
```

10.33 UPCA

```
object UPCA {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setReportCheckDigit(enable: Boolean): SettingCommand

    @JvmStatic
    fun setPreamble(value: UpcAType.PreambleType): SettingCommand

    @JvmStatic
```

```
    fun setSupplemental2Bit(enable: Boolean): SettingCommand

    @JvmStatic
    fun setSupplemental5Bit(enable: Boolean): SettingCommand
}
```

10.33.1 UpcAType

```
object UpcAType {
    enum class PreambleType : TypeValue {
        NONE, SYS_CHAR, COUNTRY_SYS_CHAR
    }
}
```

10.34 UPCE1

```
object UPCE1 {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setConvertToUpcA(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setPreamble(value: UpcE1Type.PreambleType): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setReportCheckDigit(enable: Boolean): SettingCommand
}
```

10.34.1 UpcE1Type

```
object UpcE1Type {
    enum class PreambleType : TypeValue {
        NONE, SYS_CHAR, COUNTRY_SYS_CHAR
    }
}
```

10.35 UPCEan

```
object UPCEan {
    @JvmStatic
    fun setReportEan8CheckDigit(enable: Boolean): SettingCommand

    @JvmStatic
    fun setReportEan13CheckDigit(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setSupplementalMode(value: UpcEanType.SupplementalModeType): SettingCommand

```

```
    @JvmStatic //E4770 is not supported
    fun setSupplementalAimIdFormat(value: UpcEanType.SupplementalAimIdFormatType):
SettingCommand

    @JvmStatic //E4770 is not supported
    fun setReducedQuietZone(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setBookland(value: UpcEanType.BooklandType): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setUccCouponExtend(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setCouponReport(value: UpcEanType.CouponReportType): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setIssnEan(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setTranslateUpcAToEan13(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setSupplementalRedundancy(value: Int): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setEanSupplemental2(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setEanSupplemental5(enable: Boolean): SettingCommand

    @JvmStatic //SE4107, SE5500 is not supported
    fun setEan8ReadSupplementals(enable: Boolean): SettingCommand

    @JvmStatic //SE4107, SE5500 is not supported
    fun setEan13ReadSupplementals(enable: Boolean): SettingCommand

    @JvmStatic //SE4107, SE5500 is not supported
    fun setEan8Supplemental2(enable: Boolean): SettingCommand

    @JvmStatic //SE4107, SE5500 is not supported
    fun setEan13Supplemental2(enable: Boolean): SettingCommand

    @JvmStatic //SE4107, SE5500 is not supported
    fun setEan8Supplemental5(enable: Boolean): SettingCommand

    @JvmStatic //SE4107, SE5500 is not supported
    fun setEan13Supplemental5(enable: Boolean): SettingCommand
}
```

- **setEan8Supplemental2(enable)** and **setEan8Supplemental5(enable)** work only if **setEan8ReadSupplementals(true)** is called beforehand.

- **setEan13Supplemental2(enable)** and **setEan13Supplemental5(enable)** work only if **setEan13ReadSupplementals(true)** is called beforehand.

10.35.1 UpcEanType

```
object UpcEanType {
    enum class SupplementalModeType : TypeValue {
        IGNORE_SUPPLEMENTALS,
        DECODE_WITH_SUPPLEMENTALS,
        AUTODISCRIMINATE_SUPPLEMENTALS,
        SMART_SUPPLEMENTAL_MODE,
        ENABLE_378_379,
        ENABLE_978_979,
        ENABLE_414_419_434_439,
        ENABLE_977,
        ENABLE_491,
        USER_PROGRAMMABLE_TYPE1,
        USER_PROGRAMMABLE_TYPE1_2,
        SMART_PLUS_USER_PROGRAMMABLE1,
        SMART_PLUS_USER_PROGRAMMABLE1_2
    }

    enum class SupplementalAimIdFormatType : TypeValue {
        SEPARATE, COMBINED, SEPARATE_TRANSMISSION
    }

    enum class CouponReportType : TypeValue {
        OLD_FORMAT, NEW_FORMAT, AUTODISCRIMINATE
    }

    enum class BooklandType : TypeValue {
        ISBN10, ISBN13
    }
}
```

10.36 UPCE

```
object UPCE {
    @JvmStatic
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setConvertToUpcA(enable: Boolean): SettingCommand

    @JvmStatic
    fun setPreamble(value: UpcEType.PreambleType): SettingCommand

    @JvmStatic
    fun setReportCheckDigit(enable: Boolean): SettingCommand

    @JvmStatic
    fun setSupplemental2Bit(enable: Boolean): SettingCommand

    @JvmStatic
    fun setSupplemental5Bit(enable: Boolean): SettingCommand
}
```

10.36.1 UpcEType

```
object UpcEType {
    enum class PreambleType : TypeValue {
        NONE, SYS_CHAR, COUNTRY_SYS_CHAR
    }
}
```

10.37 UpuFicsPostal

```
object UpuFicsPostal {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.38 USPlanet

```
object USPlanet {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand

    @JvmStatic //E4770 is not supported
    fun setReportCheckDigit(enable: Boolean): SettingCommand
}
```

10.39 USPostnet

```
object USPostnet {
    @JvmStatic //E4770 is not supported
    fun setEnable(enable: Boolean): SettingCommand
}
```

10.40 General

```
object General {
    @JvmStatic
    fun findMe(): SettingCommand

    @JvmStatic
    fun setAimer(enable: Boolean): SettingCommand

    @JvmStatic
    fun setIllumination(enable: Boolean): SettingCommand

```

```
    @JvmStatic
    fun setSound(type: SoundType): SettingCommand

    @JvmStatic
    fun setVibrate(enable: Boolean): SettingCommand

    @JvmStatic
    fun setLedEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setButtonEnable(enable: Boolean): SettingCommand

    @JvmStatic
    fun setWithCodeType(enable: Boolean): SettingCommand
}
```

10.40.1 SoundType

```
enum class SoundType : TypeValue {
    ON, OFF, MUTE_ON_FAILED_SCAN
}
```

10.41 ReaderParams

```
object ReaderParams {
    @JvmStatic
    fun setReadMode(value: ReaderType.ReadModeType): SettingCommand

    /** value is in deciseconds (seconds × 10). e.g., 1s -> 10, 1.5s -> 15, 10s -> 100. */
    @JvmStatic
    fun setLaserOnTime(value: Int): SettingCommand

    @JvmStatic
    fun setInverse1D(value: ReaderType.Inverse1DType): SettingCommand

    @JvmStatic
    fun setQuietZoneLevelFor1D(value: ReaderType.QuietZoneLevelType): SettingCommand

    @JvmStatic
    fun setPoorQualityDecodeEffort(value: ReaderType.PoorQualityDecodeEffortType):
                                                                        SettingCommand
}
```

10.41.1 ReaderType

```
object ReaderType {
    enum class Inverse1DType : TypeValue {
        REGULAR_ONLY, INVERSE_ONLY, INVERSE_AUTODETECT
    }

    enum class QuietZoneLevelType : TypeValue {
        LEVEL_0, LEVEL_1, LEVEL_2, LEVEL_3
    }

```

```
    enum class PoorQualityDecodeEffortType : TypeValue {
        LEVEL_0, LEVEL_1, LEVEL_2, LEVEL_3
    }

    enum class ReadModeType : TypeValue {
        ASYNC, SYNC
    }
}
```

10.42 BasicDataFormat

```
object BasicDataFormat {
    @JvmStatic
    fun setTransmitCodeID(value: BasicFormatType.TransmitCodeIdType): SettingCommand

    @JvmStatic
    fun setEndChar(value: BasicFormatType.EndCharType): SettingCommand

    @JvmStatic
    fun setSubString(value: String): SettingCommand

    @JvmStatic
    fun setRemoveFnc(enable: Boolean): SettingCommand

    @JvmStatic
    fun setTranslateData(value: String): SettingCommand

    @JvmStatic
    fun setPrefixPostfixAsAsciiHex(enable: Boolean): SettingCommand

    @JvmStatic
    fun setPrefix(value: String): SettingCommand

    @JvmStatic
    fun setPostfix(value: String): SettingCommand
}
```

10.42.1 BasicFormatType

```
object BasicFormatType {
    enum class TransmitCodeIdType : TypeValue {
        NONE, AIM, SYMBOL
    }

    enum class EndCharType : TypeValue {
        ENTER, TAB, NONE
    }
}
```

10.42.2 String Value Format

All string values must follow the format below. Be sure to include special characters.

- setSubstring(String)

The scanner returns data truncated according to the configured indices.

Input format: start index,end index.

ex) 3,5

- setTranslateData(String)

Maps the output bytes from the scanner by converting original bytes to replacement bytes.

Input format: [original value,replacement value];[original value,replacement value]... up to 8 pairs.

ex) 5B,28;5D,29

- setPrefix(String)

The configured bytes are added as a prefix to the scanner output.

Input format: up to 16 characters (16 bytes).

Ex) abcdef

- setPostfix(String)

The configured bytes are appended as a suffix to the scanner output.

Input format: up to 16 characters (16 bytes).

Ex) abcdef

10.43 Dev

```
1. object Dev {
2.     @JvmStatic
3.     fun setBatteryCallback(enable: Boolean): SettingCommand
4. }
```

11. M3Utils

These functions are useful when developing applications with the WR15 SDK.

1. toUtf8String

```
/* M3Utils */

@JvmOverloads
@JvmStatic
fun toUtf8String(bytes: ByteArray, charset: Charset = Charsets.UTF_8): String
```

Decodes a byte array using the specified charset and returns a String. The default is UTF-8.

2. isDecimalPair

```
/* M3Utils */

@JvmStatic
fun isDecimalPair(input: String): Boolean
```

Validates whether the input is a string in the **"Int,Int"** format.
Use this for parameter validation when calling **Settings.BasicDataFormat.setSubString(String)**.

3. sHexPairsSequenceAndMaxLength8

```
/* M3Utils */

@JvmStatic
fun isHexPairsSequenceAndMaxLength8(value: String): Boolean
```

Validates whether the input is a string in the **"hh,hh;hh,hh"** format and contains at most 8 pairs separated by semicolons.
Use this for parameter validation when calling **Setting.BasicDataFormat.setTranslateData(String)**.

4. generateQrBitmap(DeviceState, Int)

```
/* M3Utils */

@JvmOverloads
@JvmStatic
fun generateQrBitmap(deviceState: DeviceState, size: Int = 512): Bitmap?
```

Generate a WR15 configuration QR code using **DeviceState**.

- deviceState: DeviceState: Object to encode as a QR code.
- size: Int: Size of the QR code (default: 512).

12. ERROR CODE

The SDK provides a variety of error codes that correspond to different failure cases.

12-1 TransportErrorCatalog

```
enum class TransportErrorCatalog(val code: Int, val msg: String) {
    INVALID_COMMAND(1001, "Invalid command"),
    CONNECTION_FAILED(1002, "Connection failed"),
    UNSUPPORTED_SETTING(1003, "Unsupported setting"),
    SDK_ERROR(1004, "SDK internal error"),
    SETTING_FAILED(1005, "Setting request failed"),
    REQUEST_TIMEOUT(1006, "Request timed out"),
    UNKNOWN_ERROR(1100, "Unknown error occurred"),
}
```

12-2 ConnectErrorCatalog

```
enum class ConnectErrorCatalog(val code: Int, val msg: String) {
    ALREADY_CONNECTING(2001, "Already connecting to a device."),
    CONNECTION_FAILED(2004, "Connection failed."),
    DEVICE_SETTINGS_LOAD_FAILED(2005, "Failed to load Device Settings"),
    UNKNOWN(2100, "Unknown connection error"),
}
```