CipherLab
User Guide

1800 Software Development Kit

.NET Programming

Version 2.40

CIPHERLAB

CIPHERLAB CO., LTD.
Website: http://www.cipherlab.com

RELEASE NOTES

Version	Date	Notes
2.40	Nov. 07, 2014	<Modification> ● Updated the following methods: ✧ RFIDDirectStartInventoryRound() ✧ RFIDReadTagByEPC() ✧ RFIDReadTagByTID()
2.30	Jul. 08, 2014	<Modification> ● Renamed Serial Port Connection Service to 1800 EZSuite USB Service
2.20	Mar. 25, 2014	<New> ● Added the following: ✧ Pre-requisites ✧ 1.2 Using the Bluetooth Virtual COM Port Connector ✧ 4.2 BluetoothVCPConnector Class <Modification> ● Adjusted the order of the original sections under Chapter 1 and 4 ● Removed Appendix I
2.10	Jan. 24, 2013	<New> ● Added the following methods: ✧ RFIDDirectCancelInventoryRound() ✧ RFIDDirectKillTag() ✧ RFIDDirectLockTag() ✧ RFIDDirectReadTagByEPC() ✧ RFIDDirectReadTagByTID() ✧ RFIDDirectPermanentLockTag() ✧ RFIDDirectStartInventoryRound() ✧ RFIDDirectUnlockTag() ✧ RFIDDirectWriteTagByEPC() ✧ RFIDDirectWriteTagByTID() ✧ RFIDWriteTagEPCBank() ✧ RFIDWriteTagPassword() <Modification> ● Renamed EnableScanner() to EnableDeviceTrigger() ● Changed Return Value type and added parameter to ReadDeviceMemoryData()
2.00	Nov. 14, 2012	Changed document structure
1.00	Aug. 21, 2012	Initial release

CONTENTS

INTRODUCTION

The 1800 Series .NET Software Development Kit provides a set of APIs for configuring the 1800 Series RFID devices, as well as reading and writing RFID tags. An application programmer can easily develop his or her application program through the calls of the APIs. CipherLab provides two versions of the SDK: Windows and Windows Mobile.

The contents of each version are shown below. Please make sure the files are completely included.

Windows version:
- CipherLab.RFID.Device1800.Config.dll
- CipherLab_VCOM_CDCx64_V1.01.msi
- CipherLab_VCOM_CDCx86_V1.01.msi
- CP210x_VCP_Win2K_XP_S2K3_V5.40.24.exe
- InTheHand.Net.Personal.dll
- SerialPortConnection.exe
- SerialPortConnection.reg
- SerialPort.dll

Windows Mobile version:
- CipherLab.Bluetooth.dll
- CipherLab.RFID.Device1800.Config.dll

Before start using the library, we strongly recommend that you read this document thoroughly and keep it at hand for quick reference.

Thank you for choosing CipherLab products!

Development Tool

It is assumed that the programmer has prior knowledge of Windows programming. For information on Windows application programming interface (API), essential online resource is available on Microsoft website –

http://msdn2.microsoft.com/en-us/library/default.aspx

The development environment and tools can be one of the following choices:

- Visual Studio 2008
- Visual Studio 2005

In order to use the SDK for your application software development, you need to reference your project to **CipherLab.RFID.Device1800.Config.DLL**.

Use the **CipherLab.RFID.Device1800.Config** namespace in your project to access the library's classes, structures, enumerations, and methods without specifying fully qualified names.

```
using CipherLab.RFID.Device1800.Config;
```

Pre-requisites

The Windows version of the SDK provides three connection methods for your application program to communicate with the 1800 Series device: Bluetooth socket, Bluetooth virtual COM port and USB.

The Windows Mobile version only provides Bluetooth socket connection.

Preparing for Bluetooth Socket Connection

No extra setup is required when using Bluetooth socket connection. Application programmers only need to reference to the configuration library in their .NET projects in order to use the APIs. However, the following Bluetooth library must also be present since the configuration library relies on it for communication:

Windows version:
- InTheHand.Net.Personal.dll

Windows Mobile version:
- CipherLab.Bluetooth.dll

Preparing for Bluetooth Virtual COM Port Connection (Windows Only)

Before using the Bluetooth virtual COM port connection, your 1800 Series device must be paired with your Windows PC. Please refer to the 1800 user manual for the details on how to pair your 1800 Series device with your Windows PC. The serial port component library (SerialPort.dll) must be present since the SDK relies on it for Bluetooth virtual COM port connection.

Preparing 1800 EZSuite USB Service for USB Connection (Windows Only)

Communication through USB is handled by a Windows service called "1800 EZSuite USB Service." This Windows service watches for the USB cable connection of your 1800 Series device and prepares for device communication. Therefore, you must set up the 1800 EZSuite USB Service if you plan to use USB as a connection option in your application program. We strongly recommend that you read this section so that you know which files to include and which Registry key and values to create in your application program installer

Note: If you have already installed CipherLab software products that utilize the 1800 EZSuite USB Service, such as 1800 EZSuite, you do not need to perform the instructions described in this section.

The 1800 EZSuite USB Service relies on the serial port component library (SerialPort.dll) and one of the included USB virtual COM port drivers for device communication. Therefore, you must install one of these drivers onto your computer for the 1800 EZSuite USB Service to work properly. The drivers included are:

File	Description
CipherLab_VCOM_CDCx86_V1.01.msi	CipherLab VCOM CDC driver for 32-bit Windows
CipherLab_VCOM_CDCx64_V1.01.msi	CipherLab VCOM CDC driver for 64-bit Windows
CP210x_VCP_Win2K_XP_S2K3_V5.40.24.exe	Silicon Labs Virtual COM Port driver

After the driver is installed, we can go on to install the 1800 EZSuite USB Service. Copy the file "SerialPortConnection.exe" to a desired location on your hard drive. Then, execute the following command in the Windows Command Prompt to install the service:

```
SerialPortConnection.exe –install
```

To uninstall the service, use the following command:

```
SerialPortConnection.exe –uninstall
```

The 1800 EZSuite USB Service recognizes the following DWORD values in the "HKEY_LOCAL_MACHINE\SOFTWARE\CipherLab\SerialPortConnectionService" Registry key:

Name	Value	Description
KeepAlive	1	When this value is set to 1, the 1800 EZSuite USB Service will prevent the 1800 Series device from entering power-saving mode.
Port	6453	The port used by the 1800 EZSuite USB Service to communicate with the SDK.
Driver	0	When this value is set to 1 the 1800 EZSuite USB Service will watch for 1800 Series device using CipherLab VCOM CDC driver exclusively. When this value is set to 2 the 1800 EZSuite USB Service will watch for 1800 Series device using Silicon Labs Virtual COM Port driver exclusively. When this value is set to 0 the 1800 EZSuite USB Service will watch for 1800 Series device using either CipherLab or Silicon Labs driver.

If the above values do not exist, the 1800 EZSuite USB Service will automatically create them for you on its first start up. If you would like to create these Registry value forehand, double-click on "SerialPortConnection.reg" to create them. In addition, the 1800 EZSuite USB Service automatically creates a "SerialPort" string value when an 1800 Series device is successfully connected through USB. This string value is automatically deleted when the 1800 Series device is disconnected. This value becomes handy if your application program needs to know which COM port number is assigned to the connected 1800 Series device.

Note: The "SerialPort" string value is different from the "Port" DWORD value. The former is the COM port which an 1800 Series device is connected to. The latter is the port used by the 1800 EZSuite USB Service to communicate with the SDK.

Next, you need to start the 1800 EZSuite USB Service. In the Windows Command Prompt, type the following:

```
net start “1800 EZSuite USB Service”
```

Alternatively, you can start the 1800 EZSuite USB Service either by rebooting your computer or using the Windows Administrative Tools.

Now, you are all set. Once the service is started, your application is able to communicate with the connected 1800 Series device through the 1800 EZSuite USB Service. Please proceed to “**Chapter 1: Connecting to an 1800 Series Device**”.

Chapter 1

CONNECTING TO AN 1800 SERIES DEVICE

The 1800 Series .NET SDK uses a component called "connector" to establish the connection between an 1800 Series device and a Windows PC/Windows Mobile terminal. The connector provides methods for your application software to connect to and disconnect from the 1800 Series device. The connector also provides a method for the application program to check the current connection status. Additionally, you can assign callback methods to the connector such as connect callback method, disconnect callback method, event callback method, etc. Therefore, the application program will be notified when the 1800 Series device is connected, gets disconnected, or raises an event.

The Windows version of the library provides three connectors: Bluetooth Socket Connector, Bluetooth Virtual COM Port Connector, and Serial Port Connector (USB). To connect the 1800 Series device with a Windows Mobile terminal you must use the Windows Mobile version of the library. The Windows Mobile version of the library provides Bluetooth socket connection only. Please refer to the section "Using the Bluetooth Socket Connector" for how to use the Bluetooth Socket Connector.

In This Chapter

1.1 Using the Bluetooth Socket Connector

To use the Bluetooth Socket Connector you must instantiate it first:

```
Connector connector;
connector = new BluetoothSocketConnector();
```

The Bluetooth Socket Connector provides four delegates for you to optionally define callback methods: ConnectDelegate, DisconnectDelegate, EventDelegate, and DataDelegate. The method associated with ConnectDelegate will be invoked when the connector establishes a connection to an 1800 Series device. The method associated with DisconnectDelegate is invoked once the connected 1800 Series device is disconnected from the connector. The method associated with EventDelegate will be invoked when an event is received from the connected 1800 Series device. See *2.8 DeviceEvent* for a list of events. The method associated with DataDelegate is invoked when a RFID tag is read by the connected 1800 Series device.

```
connector.ConnectDelegate += OnConnected;
connector.DisconnectDelegate += OnDisconnected;
connector.EventDelegate += OnNotificationEvent;
connector.DataDelegate += OnDataReceived;

private void OnConnected()
{
    // do something when connected
}

private void OnDisconnected()
{
    // do something when disconnected
}

private void OnNotificationEvent(DeviceEvent mask)
{
   // handle events
   if ((mask & DeviceEvent.ExtModeChange) == DeviceEvent.ExtModeChange)
   {
       // handle ALTERNATE_MODE event
   }
}

Private void OnDataReceived(EPCData data)
{
    // process scanned data
}
```

In the Windows version of the library, you must search for the 1800 Series device before you can connect to it. To search for 1800 Series device, call the connector's DiscoverDevice() method:

```
BTDevice[] devices = connector.DiscoverDevice();
```

The Bluetooth Socket Connector in the Windows Mobile version of library provides a GetPairedDevice() method that gives you a list of 1800 Series devices that have been paired before.

```
BTDevice[] devices = connector.GetPairedDevice();
```

Before connecting, assign the name of the device you wish to connect to the Bluetooth Socket Connector. If the device requires a Bluetooth pin code set it as well.

```
connector.device = devices[0];
connector.pincode = "1234";
```

Finally, call Bluetooth Socket Connector's Connect() method to connect:

```
connector.Connect();
```

Once the device is connected instantiate the configurator object. This can be done in the connect callback method.

```
private void OnConnected()
{
    RFIDConfigurator = new Configurator(connector);
}
```

Once the configurator object is instantiated, you can use its methods to update the device configuration.

To disconnect the 1800 Series device, simply call the Bluetooth Socket Connector's Disconnect() method:

```
connector.Disconnect();
```

To check the connection status, use the Bluetooth Socket Connector's IsConnected() method:

```
if (connector.IsConnected() == false)
{
    // do something
}
```

1.2 Using the Bluetooth Virtual COM Port Connector (Windows Only)

To use the Bluetooth Virtual COM Port Connector you must instantiate it with a given COM port first:

```
Connector connector;
connector = new BluetoothVCPConnector("COM17");
```

Note: Your 1800 Series device must be paired with your Windows PC before you can use the Bluetooth Virtual COM Port Connector.

The Bluetooth Virtual COM Port Connector provides four delegates for you to optionally define callback methods: ConnectDelegate, DisconnectDelegate, EventDelegate, and DataDelegate. The method associated with ConnectDelegate will be invoked when the connector establishes a connection to an 1800 Series device. The method associated with DisconnectDelegate is invoked once the connected 1800 Series device is disconnected from the connector. The method associated with EventDelegate will be invoked when an event is received from the connected 1800 Series device. See *2.8 DeviceEvent* for a list of events. The method associated with DataDelegate is invoked when a RFID tag is read by the connected 1800 Series device.

```
connector.ConnectDelegate += OnConnected;
connector.DisconnectDelegate += OnDisconnected;
connector.EventDelegate += OnNotificationEvent;
connector.DataDelegate += OnDataReceived;

private void OnConnected()
{
    // do something when connected
}

private void OnDisconnected()
{
    // do something when disconnected
}

private void OnNotificationEvent(DeviceEvent mask)
{
   // handle events
   if ((mask & DeviceEvent.ExtModeChange) == DeviceEvent.ExtModeChange)
   {
       // handle ALTERNATE_MODE event
   }
}

Private void OnDataReceived(EPCData data)
{
    // process scanned data
}
```

To connect to the 1800 Series device, call the Bluetooth Virtual COM Port Connector's Connect method with a given port name:

```
connector.Connect();
```

Once the device is connected instantiate the configurator object. This can be done in the connect callback method.

```
private void OnConnected()
{
    RFIDConfigurator = new Configurator(connector);
}
```

Once the configurator object is instantiated, you can use its methods to update the device configuration.

To disconnect the 1800 Series device, simply call the connector's Disconnect() method:

```
connector.Disconnect();
```

To check the connection status, use the serial port connector's IsConnected() method:

```
if (connector.IsConnected() == false)
{
    // do something
}
```

1.3 Using the Serial Port Connector (Windows Only)

To use the Serial Port Connector you must instantiate it first:

```
Connector connector;
connector = new SerialPortConnector();
```

The Serial Port Connector provides four delegates for you to optionally define callback methods: ConnectDelegate, DisconnectDelegate, EventDelegate, and DataDelegate. The method associated with ConnectDelegate will be invoked when the connector establishes a connection to an 1800 Series device. The method associated with DisconnectDelegate is invoked once the connected 1800 Series device is disconnected from the connector. The method associated with EventDelegate will be invoked when an event is received from the connected 1800 Series device. See *2.8 DeviceEvent* for a list of events. The method associated with DataDelegate is invoked when a RFID tag is read by the connected 1800 Series device.

```
connector.ConnectDelegate += OnConnected;
connector.DisconnectDelegate += OnDisconnected;
connector.EventDelegate += OnNotificationEvent;
connector.DataDelegate += OnDataReceived;

private void OnConnected()
{
    // do something when connected
}

private void OnDisconnected()
{
    // do something when disconnected
}

private void OnNotificationEvent(DeviceEvent mask)
{
   // handle events
   if ((mask & DeviceEvent.ExtModeChange) == DeviceEvent.ExtModeChange)
   {
       // handle ALTERNATE_MODE event
   }
}

Private void OnDataReceived(EPCData data)
{
    // process scanned data
}
```

To connect to the 1800 Series device through USB, call one of the Serial Port Connector's connect method:

```
connector.Connect();
```

Once the device is connected instantiate the configurator object. This can be done in the connect callback method.

```
private void OnConnected()
{
    RFIDConfigurator = new Configurator(connector);
}
```

Once the configurator object is instantiated, you can use its methods to update the device configuration.

To disconnect the 1800 Series device, simply call the Serial Port Connector's Disconnect() method:

```
connector.Disconnect();
```

To check the connection status, use the Serial Port Connector's IsConnected() method:

```
if (connector.IsConnected() == false)
{
    // do something
}
```

Chapter 2

ENUMERATIONS

An enumeration is a special type in the .NET Framework that is comprised of a number of named constants just like a header, item and others. Here, you can program 1800 Series RFID Reader into some enumerations with a prefix "public enum" as listed below.

In This Chapter

2.1 BeepDuration

The **BeepDuration** enumeration defines the beep duration of the 1800 device beeper.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum BeepDuration
{
        Shortest,
        Short,
        Long,
        Longest,
}
```

Shortest
Shortest duration

Short
Short duration

Long
Long duration

Longest
Longest duration

See Also

3.2 DeviceNotification
Configurator.GetDeviceNotificationState Method
Configurator.SetDeviceNotificationState Method

2.2 BeepFrequency

The **BeepFrequency** enumeration defines the beep frequency of the 1800 device beeper.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum BeepFrequency
{
        Freq_8kHz,
        Freq_4kHz,
        Freq_2kHz,
        Freq_1kHz,
}
```

Freq_8kHz
Beep frequency 8kHz

Freq_4kHz
Beep frequency 4kHz

Freq_2kHz
Beep frequency 2kHz

Freq_1kHz
Beep frequency 1kHz

See Also

3.2 DeviceNotification
Configurator.GetDeviceNotificationState Method
Configurator.SetDeviceNotificationState Method

2.3 BeepVolume

The **BeepVolume** enumeration defines the volume of the 1800 device beep.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum BeepVolume
{
        Mute,
        Low,
        Medium,
        High,
}
```

Mute
Turn off the beep

Low
Low beep volume

Medium
Medium beep volume

High
High beep volume

See Also

3.2 DeviceNotification
Configurator.GetDeviceNotificationState Method
Configurator.SetDeviceNotificationState Method

2.4 BluetoothInterface

The **BluetoothInterface** enumeration defines *Bluetooth*® connection type of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum BluetoothInterface
{
        SPP_Slave=0,
        SPP_Master=3,
        HID=5,
        BT3610=6,
}
```

SPP_Slave
Connection interface Bluetooth® SPP Slave

SPP_Master
Connection interface Bluetooth® SPP Master

HID
Connection interface Bluetooth® HID

BT3610
Connection interface CipherLab 3610 dongle

See Also

Configurator.GetBluetoothInterfaceType Method
Configurator.SetBluetoothInterfaceType Method

2.5 CapsLockState

The **CapsLockState** enumeration defines the caps lock state of the selected HID keyboard interface.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum CapsLockState
{
        Off,
        On,
        Auto,
}
```

Off

Caps Lock off

On

Caps Lock on

Auto

Device automatically detects the Caps Lock status before data is transmitted

See Also

3.4 HID Configurator.GetConnectionInterface3610Hid Method
Configurator.GetConnectionInterfaceHid Method
Configurator.SetConnectionInterface3610Hid Method
Configurator.SetConnectionInterfaceHid Method

2.6 DataPortal

The **DataPortal** enumeration defines the outlet of scanned EPC data.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum DataPortal
{
        USB,
        Bluetooth,
}
```

USB

Output scanned data to USB Virtual COM port

Bluetooth

Output scanned data to Bluetooth

See Also

Configurator.GetDataPortal Method
Configurator.SetDataPortal Method

2.7 DataSequenceSection

The **DataSequenceSection** enumeration defines the fields of output data.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum DataSequenceSection
{
        NotUsed,
        Count,
        Timestamp,
        Data
}
```

NotUsed
Not used

Count
Include tag count in the output data

Timestamp
Include timestamp in the output data

Data
Include EPC data in the output data

See Also

2.25 RFIDDataField
2.30 TimestampField
3.6 RFIDData
3.8 Timestamp
Configurator.GetCounterFormat Method (byte[], byte[], char)
Configurator.GetCounterFormat Method (string, string, char)
Configurator.GetOutputDataSequence
Configurator.GetRFIDDataFormat Method (byte[], byte[], RFIDData)
Configurator.GetRFIDDataFormat Method (string, string, RFIDData)
Configurator.GetTimestampFormat Method (byte[], byte[], Timestamp)
Configurator.GetTimestampFormat Method (string, string, Timestamp)
Configurator.SetCounterFormat Method (byte[], byte[], char)
Configurator.SetCounterFormat Method (string, string, char)
Configurator.SetOutputDataSequence
Configurator.SetRFIDDataFormat Method (byte[], byte[], RFIDData)
Configurator.SetRFIDDataFormat Method (string, string, RFIDData)
Configurator.SetTimestampFormat Method (byte[], byte[], Timestamp)
Configurator.SetTimestampFormat Method (string, string, Timestamp)

2.8 DeviceEvent

The **DeviceEvent** enumeration defines the events raised by the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
[Flags]
public enum DeviceEvent
{
        PowerSavingMode = 0x0001,
        DeviceShutdown = 0x0002,
        BluetoothDisconnect = 0x0004,
        SettingChange = 0x0008,
        LowBattery = 0x0010,
        ExtModeChange = 0x0020,
        ScannerFailure = 0x0040,
        SingleTagNoTagFound = 0x0080,
        MultiTagScanComplete = 0x0100,
        All = PowerSavingMode | DeviceShutdown | BluetoothDisconnect
              | SettingChange | LowBattery | ExtModeChange |
              ScannerFailure | SingleTagNoTagFound |
              MultiTagScanComplete,
};
```

PowerSavingMode
Entering power saving mode.

DeviceShutdown
1800 device shutting down.

BluetoothDisconnect
Bluetooth® disconnecting.

SettingChange
1800 device setting changed.

LowBattery
Low battery power. This event is raised once every 30 seconds when if the battery is not being charged.

ExtModeChange
RFID/EXT. switch position changed.

ScannerFailure
Scanner fails to initialize or not responding to tag scan.

SingleTagNoTagFound
No tag found during a single tag scan session.

MultiTagScanComplete
Multi-tag scan session completed.

All
All of the above combined.

2.9 DeviceResponse

The **DeviceResponse** enumeration defines the status responded by 1800 device when calling the RFIDDirect methods.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum DeviceResponse
{
        OperationSuccess = 0,
        OperationFail,
        DeviceTimeOut,
        DeviceBusy
}
```

OperationSuccess
Operation succeeded

OperationFail
Operation failed

DeviceTimeOut
1800 device timed out.

DeviceBusy
1800 device is busy.

See Also

Configurator.RFIDDirectKillTag Method
Configurator.RFIDDirectLockTag Method
Configurator.RFIDDirectReadTagByEPC Method
Configurator.RFIDDirectReadTagByTID Method
Configurator.RFIDDirectPermanentLockTag Method
Configurator.RFIDDirectStartInventoryRound Method
Configurator.RFIDDirectUnlockTag Method
Configurator.RFIDDirectWriteTagByEPC Method
Configurator.RFIDDirectWriteTagByTID Method

2.10 DigitPosition

The **DigitPosition** enumeration defines the digit position of selected HID keyboard interface.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum DigitPosition
{
        Normal,
        LowerRow,
        UpperRow,
}
```

Normal
Digit position depends on the [Shift] key or [Shift Lock] settings

LowerRow
QWERTY or QWERTZ keyboard

UpperRow
AZERTY keyboard

See Also

3.4 HID Configurator.GetConnectionInterface3610Hid Method
Configurator.GetConnectionInterfaceHid Method
Configurator.SetConnectionInterface3610Hid Method
Configurator.SetConnectionInterfaceHid Method

2.11 EPCEncodingScheme

The **EPCEncodingScheme** enumeration defines the EPC encoding schemes that the 1800 device recognizes and/or filters.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum EPCEncodingScheme
{
        None = 0x0000,
        GDTI96 = 0x0001,
        GSRN96 = 0x0002,
        USDoD96 = 0x0004,
        SGTIN96 = 0x0008,
        SSCC96 = 0x0010,
        SGLN96 = 0x0020,
        GRAI96 = 0x0040,
        GIAI96 = 0x0080,
        GID96 = 0x0100,
        SGTIN198 = 0x0200,
        GRAI170 = 0x0400,
        GIAI202 = 0x0800,
        SGLN195 = 0x1000,
        GDTI113 = 0x2000,
        ADI = 0x4000,
}
```

None
None of the defined encoding schemes is in use

GDTI96
96-bit "Global Document Type Identifier" encoding scheme

GSRN96
96-bit "Global Service Relation Number" encoding scheme

USDoD96
96-bit "US Department of Defense Identifier" encoding scheme

SGTIN96
96-bit "Serialized Global Trade Item Number" encoding scheme

SSCC96
96-bit "Serial Shipping Container Code" encoding scheme

SGLN96
96-bit "Global Location Number With or Without Extension" encoding scheme

GRAI96
96-bit "Global Returnable Asset Identifier" encoding scheme

GIAI96
96-bit "Global Individual Asset Identifier" encoding scheme

GID96

96-bit "General Identifier" encoding scheme

SGTIN198

198-bit "Serialized Global Trade Item Number" encoding scheme

GRAI170

170-bit "Global Returnable Asset Identifier" encoding scheme

GIAI202

202-bit "Global Individual Asset Identifier" encoding scheme

SGLN195

195-bit "Global Location Number With or Without Extension" encoding scheme

GDTI113

113-bit "Global Document Type Identifier" encoding scheme

ADI

"Aerospace and Defense Identifier" encoding scheme

See Also

3.3 EPCFilter
Configurator.GetRecognizedEPCEncoding
Configurator.SetRecognizedEPCEncoding

2.12 EPCFilterType

The **EPCFilterType** enumeration defines the type of EPC filter. The 1800 device supports inclusion and exclusion filters.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum EPCFilterType
{
        None,
        IncludePattern,
        ExcludePattern,
}
```

None
Filter not in use

IncludePattern
Include EPC

ExcludePattern
Exclude EPC

See Also

3.3 EPCFilter
Configurator.GetEPCFilter
Configurator.SetEPCFilter

2.13 FilterMethod

The **FilterMethod** enumeration defines how EPC should be filtered by the 1800 device upon reading tags.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum FilterMethod
{
        FilterValue,
        Partition1,
        Partition2,
}
```

FilterValue
Filter EPC by its filter value

Partition1
Filter EPC by its first partition

Partition2
Filter EPC by its second partition

See Also

3.3 EPCFilter
Configurator.GetEPCFilter
Configurator.SetEPCFilter

2.14 FirmwareUpdateInterface

The **FirmwareUpdateInterface** enumeration defines the connection interface for updating the 1800 device firmware.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum FirmwareUpdateInterface
{
        Current,
        Bluetooth,
        USB,
}
```

Current
Use the current interface to upgrade firmware.

Bluetooth
Use Bluetooth® to upgrade firmware.

USB
Use USB to upgrade firmware.

See Also

Configurator.SetFirmwareUpdateInterface

2.15 HIDCharTransmitMode

The **HIDCharTransmitMode** enumeration defines the character transmission mode of the selected HID keyboard interface.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum HIDCharTransmitMode
{
        BatchProcessing,
        ByCharacter,
}
```

BatchProcessing

Send data to PC in batch.

ByCharacter

Send data to PC, one character at a time.

See Also

3.4 HID
Configurator.GetConnectionInterface3610Hid Method
Configurator.GetConnectionInterfaceHid Method
Configurator.SetConnectionInterface3610Hid Method
Configurator.SetConnectionInterfaceHid Method

2.16 HIDKeyboardType

The **HIDKeyboardType** enumeration defines the keyboard for the HID interface of the 1800 device when using *Bluetooth*® HID or BT3610 HID connection.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum HIDKeyboardType
{
        PCAT_US=64,
        PCAT_French,
        PCAT_German,
        PCAT_Italy,
        PCAT_Swedish,
        PCAT_Norwegian,
        PCAT_UK,
        PCAT_Belgium,
        PCAT_Spanish,
        PCAT_Portuguese,
        PS55A01_2_Japanese,
        PCAT_Turkish=76,
        PCAT_Hungarian,
}
```

PCAT_US
US keyboard
PCAT_French
French keyboard
PCAT_German
German keyboard
PCAT_Italy
Italian keyboard
PCAT_Swedish
Swedish keyboard
PCAT_Norwegian
Norwegian keyboard
PCAT_UK
UK keyboard
PCAT_Belgium
Belgium keyboard
PCAT_Spanish
Spanish keyboard
PCAT_Portuguese
Portuguese keyboard

PS55A01_2_Japanese
Japanese keyboard

PCAT_Turkish
Turkish keyboard.

PCAT_Hungarian
Hungarian keyboard

See Also

3.4 HID
Configurator.GetConnectionInterface3610Hid Method
Configurator.GetConnectionInterfaceHid Method
Configurator.SetConnectionInterface3610Hid Method
Configurator.SetConnectionInterfaceHid Method

2.17 InventoryType

The **InventoryType** enumeration defines the type of data returned from an inventory round.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum InventoryType
{
        EPC,
        EPC_AND_TID,
}
```

EPC
EPC only

EPC_AND_TID
Both EPC and TID

See Also
Configurator.RFIDDirectStartInventoryRound Method

2.18 KeyAction

The **KeyAction** enumeration defines the key actions of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum KeyAction
{
        None,
        TriggerDown,
        TriggerUp,
        F1Down,
        F1Up,
        F2Down,
        F2Up,
        TriggerF1Down,
        TriggerF2Down,
}
```

None
No key behavior defined

TriggerDown
Trigger key is pressed

TriggerUp
Trigger key is released

F1Down
Function key F1 is pressed

F1Up
Function key F1 is released

F2Down
Function key F2 is pressed

F2Up
Function key F2 is released

TriggerF1Down
Trigger key and F1 key are pressed simultaneously

TriggerF2Down
Trigger key and F2 key are pressed simultaneously

See Also

3.7 KeyActionString
Configurator.GetKeyActionString Method
Configurator.SetKeyActionString Method

2.19 KeyboardLayout

The **KeyboardLayout** enumeration defines the layout of selected HID keyboard interface.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum KeyboardLayout
{
        Normal,
        AZERTY,
        QWERTZ,
}
```

Normal
Standard US keyboard layout

AZERTY
AZERTY for French keyboard layout

QWERTZ
QWERTZ for German keyboard layout

See Also

3.4 HID
Configurator.GetConnectionInterface3610Hid Method
Configurator.GetConnectionInterfaceHid Method
Configurator.SetConnectionInterface3610Hid Method
Configurator.SetConnectionInterfaceHid Method

2.20 KeypadType

The **KeypadType** enumeration defines the digit transmission type of the selected HID keyboard interface.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum KeypadType
{
        AlphaNumericKeypad,
        NumericKeypad,
}
```

AlphaNumericKeypad

Alphanumeric keys on the top row of the keyboard are used for transmitting digits

NumericKeypad

Keys on the numeric keypad are used for transmitting digits

See Also

3.4 HID
Configurator.GetConnectionInterface3610Hid Method
Configurator.GetConnectionInterfaceHid Method
Configurator.SetConnectionInterface3610Hid Method
Configurator.SetConnectionInterfaceHid Method

2.21 LockTarget

The **LockTarget** enumeration defines the tag memory bank in which can be locked by the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum LockTarget
{
        KillPassword = 0,
        AccessPassword,
        EPCBank,
        TIDBank,
        UserBank
}
```

KillPassword
Kill password of the reserved bank

AccessPassword
Access password of the reserved bank

EPCBank
EPC bank

TIDBank
TID bank

UserBank
User bank

See Also

Configurator.RFIDDirectLockTag Method
Configurator.RFIDDirectPermanentLockTag Method
Configurator.RFIDDirectUnlockTag Method

2.22 MultiTagListType

The **MultiTagListType** enumeration defines the data type in which the multi-tag list of the 1800 device should store in order to avoid duplicate tag scans.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum MultiTagListType
{
        EPC,
        TID,
}
```

EPC

The 1800 device stores scanned EPCs in its multi-tag list to avoid duplicate scans

TID

The 1800 device stores tag IDs to avoid duplicate scans.

See Also

Configurator.GetMultiTagListType
Configurator.SetMultiTagListType

2.23 OperatingMode

The **Operating Mode** enumeration defines the working status of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum OperatingMode
{
        Online,
        Memory,
}
```

Online

When the 1800 device operates in online mode, it is connected to a terminal device or a computer that receives scanned data through Bluetooth® connection.

Memory

When the 1800 device operates in memory mode, it is operating without connecting to any data receiver. Scanned data is saved in its own memory.

See Also

Configurator.GetDeviceOperatingMode Method
Configurator.SetDeviceOperatingMode Method

2.24 OutputDataFormat

The **OutputDataFormat** enumeration defines the format of output data.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum OutputDataFormat
{
        PacketData,
        Hexadecimal,
        RawData,
}
```

PacketData
Output data in CipherLab proprietary packet format

Hexadecimal
Output data in hexadecimal number format.

RawData
Output data without processing

See Also

Configurator.GetOutputDataFormat
Configurator.SetOutputDataFormat

2.25 RFIDDataField

The **RFIDDataField** enumeration defines the output data fields of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum RFIDDataField
{
        NotUsed,
        CRC,
        PC,
        EPC,
        MemoryData,
        DataLength,
}
```

NotUsed
Not used

CRC
Include CRC in the output

PC
Include PC in the output

EPC
Include EPC in the output

MemoryData
Include data in the selected memory bank in the output

DataLength
Include the data length in the output

See Also

2.7 DataSequenceSection
3.6 RFIDData

2.26 RFIDMemoryBank

The **RFIDMemoryBank** enumeration defines the tag memory banks that can be selected by the 1800 device before reading or writing the tag.
Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum RFIDMemoryBank
{
        Reserved,
        EPC,
        TID,
        User,
}
```

Reserved
Reserved bank

EPC
EPC bank

TID
TID bank

User
User bank

See Also

Configurator.GetSelectedMemoryBank Method
Configurator.RFIDReadTagMassive Method
Configurator.RFIDWriteTagMassive Method
Configurator.SelectMemoryBank Method

2.27 RFIDMode

The **RFIDMode** enumeration defines read, write and inventory operations on RFID tags.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum RFIDMode
{
        Inventory,
        ReadTag,
        WriteTag,
}
```

Inventory
Inventory operation that reads RFID EPC data

ReadTag
Operation that reads data from the selected memory bank

WriteTag
Operation that writes data to the selected memory bank

See Also

Configurator.GetRFIDMode Method
Configurator.SetRFIDMode Method
Configurator.RFIDReadTagMassive Method
Configurator.RFIDWriteTagMassive Method

2.28 ScanDelay

The **ScanDelay** enumeration defines the delay interval between each tag scan.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum ScanDelay
{
        Delay_100ms,
        Delay_200ms,
        Delay_400ms,
        Delay_800ms,
        Delay_1sec,
        Delay_2sec,
        Delay_3sec,
        Delay_5sec,
}
```

Delay_100ms
Delay 100 milliseconds between each scan

Delay_200ms
Delay 200 milliseconds between each scan

Delay_400ms
Delay 400 milliseconds between each scan

Delay_800ms
Delay 800 milliseconds between each scan

Delay_1sec
Delay one second between each scan

Delay_2sec
Delay two seconds between each scan

Delay_3sec
Delay three seconds between each scan

Delay_5sec
Delay five seconds between each scan

See Also
Configurator.GetMultiTagModeSettings
Configurator.SetMultiTagModeSettings

2.29 ScanMode

The **ScanMode** enumeration defines the scanning modes of the RFID scanner.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum ScanMode
{
        SingleTag = 6,
        Test = 7,
        MultiTag = 9
}
```

SingleTag

When in "Single tag" mode, the device starts scanning for tags when the trigger is held down, and it won't stop scanning until one of the following condition is met:
(1) A tag is read.
(2) The time interval specified (1~254 sec.) expires.
(3) The trigger has been released.
By default, the time interval is set to zero, which means no time limit while scanning for tags.

Test

Do not use

MultiTag

When in "Multi-tag" mode, the device starts scanning for tags as long as the trigger is held down, and it won't stop scanning until the trigger has been released. Select an appropriate scan delay if necessary. A sequential scanning will not be started until a previous scanning is done and the scan delay time expires.
By default, it is set to read up to 128 unique tags. If tag counting is not desired, set it to 0. When the device reads a unique tag, which means the tag has never been read so far, it will save the tag info to a list used for counting. When the counter limit is reached, it will give a warning beep and discard in-coming readings.

See Also

Configurator.GetScanMode
Configurator.SetScanMode

2.30 TimestampField

The **TimestampField** enumeration defines the fields of the timestamp used in the 1800 device output data.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum TimestampField
{
        NotUsed,
        Year,
        Month,
        Day,
        DayOfWeek,
        Hour,
        Minute,
        Second,
        SecondWithMillisecond
}
```

NotUsed
Not used

Year
Include year in the timestamp

Month
Include month in the timestamp

Day
Include day in the timestamp

DayOfWeek
Include day of week in the timestamp

Hour
Include hour in the timestamp

Minute
Include minute in the timestamp

Second
Include second in the timestamp (see remarks)

SecondWithMillisecond
Include second following by millisecond in the timestamp (see remarks)

Remarks

Use *Second* or *SecondWithMillisecond* exclusively.
Using both *Second* and *SecondWithMillisecond* will cause an error.

See Also

2.7 DataSequenceSection
3.8 Timestamp

2.31 TransmissionDelay

The **TransmissionDelay** enumeration defines the delay interval between the transmission of each data record.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum TransmissionDelay
{
        Delay_0,
        Delay_250ms,
        Delay_500ms,
        Delay_1sec,
        Delay_2sec,
        Delay_3sec,
        Delay_5sec,
        Delay_8sec,
}
```

Delay_0
No delay

Delay_250ms
Delay 250 milliseconds between the transmission of each data record

Delay_500ms
Delay 500 milliseconds between the transmission of each data record

Delay_1sec
Delay one second between the transmission of each data record

Delay_2sec
Delay two seconds between the transmission of each data record

Delay_3sec
Delay three seconds between the transmission of each data record

Delay_5sec
Delay five seconds between the transmission of each data record

Delay_8sec
Delay eight seconds between the transmission of each data record

See Also
Configurator.GetOnlineModeDelay Method
Configurator.SetOnlineModeDelay Method

2.32 USBDriver

The **USBDriver** enumeration defines the USB driver software for connecting the 1800 device with the PC.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public enum USBDriver
{
        CDC=127,
        SiliconLab,
}
```

CDC

Virtual COM with Windows CDC driver

SiliconLab

Virtual COM with SiliconLab driver

See Also

Configurator.GetUSBDriver Method
Configurator.SetUSBDriver Method

Chapter 3

STRUCTURES

A structure in .NET is a composite date type consisting of number elements of other types. Here, you can program the 1800 device into some structures with a prefix "public struct" as below.

In This Chapter

3.1 DeviceInfo

The **DeviceInfo** structure contains the 1800 device system information.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public struct DeviceInfo
{
        public string Model;
        public string SerialNumber;
        public string KernelVersion;
        public string UserVersion;
        public string MacAddress;
}
```

Members

	Name	Description
public	Model	1800 device model name
public	SerialNumber	1800 device serial number
public	KernelVersion	1800 device kernel version
public	UserVersion	1800 device firmware version
public	MacAddress	1800 device *Bluetooth*® MAC address

See Also
Configurator.GetDeviceInfo

3.2 DeviceNotification

The **DeviceNotification** structure is specified for the notification settings of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public struct DeviceNotification
{
        public BeepDuration GoodReadBeeperDuration;
        public bool GoodReadBeeperEnabled;
        public BeepFrequency GoodReadBeeperFrequency;
        public int LEDDuration;
        public bool LEDEnabled;
        public bool LowBatteryAlertEnabled;
        public BeepVolume SystemBeeperVolume;
        public int VibratorDuration;
        public bool VibratorEnabled;
}
```

Members

	Name	Description
public	LowBatteryAlertEnabled	Get or set a value indicating whether the low battery alert is enabled
public	LEDEnabled	Get or set a value indicating whether the LED is enabled
public	LEDDuration	Get or set the duration of LED
public	VibratorEnabled	Get or set a value indication whether the vibrator is enabled
public	VibratorDuration	Get or set the duration of the vibrator
public	SystemBeeperVolume	Get or set the system beeper volume
public	GoodReadBeeperFrequency	Get or set the good read beeper frequency
public	GoodReadBeeperDuration	Get or set the duration of the good read beeper
public	GoodReadBeeperEnabled	Get or set a value indicating whether the good read beeper is enabled

See Also

2.1 BeepDuration
2.2 BeepFrequency,
2.3 BeepVolume
Configurator.GetDeviceNotificationState Method
Configurator.SetDeviceNotificationState Method

3.3 EPCFilter

The **EPCFilter** structure defines the condition in which tags will be filtered by the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public struct EPCFilter
{
        public bool Enabled;
        public EPCFilterType FilterType;
        public EPCEncoding Encoding;
        public FilterMethod FilterMethod;
        public int FilterValue;
        public string Part1Value;
        public string Part2FromValue;
        public string Part2ToValue;
        public int Part2MaxLength;
}
```

Members

	Name	Description
public	Enabled	A value indicating whether the 1800 device's EPC filter function is enabled.
public	FilterType	Type of filter: Inclusion or exclusion.
public	Encoding	EPC encoding scheme to filter
public	FilterMethod	How EPC is filtered: By filter value, by the first partition in EPC, or by the second partition in EPC
public	FilterValue	EPC filter value
public	Part1Value	The value of the EPC first partition to be filtered
public	Part2FromValue	The start value of the EPC second partition to be filtered
public	Part2ToValue	The end value of the EPC second partition to be filtered
public	Part2MaxLength	Maximum number of digits/characters in the EPC second partition

See Also

2.11 EPCEncodingScheme
2.12 EPCFilterType
2.13 FilterMethod
Configurator.GetEPCFilter
Configurator.SetEPCFilter

3.4 HID

The **HID** structure is specified for the settings of 1800 *Bluetooth*® HID connection.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public struct HID
{
        public HIDKeyboardType kbdType;
        public int interCharDelay;
        public CapsLockState LockState;
        public bool IgnoreCaseOnAlphabetsTransmission;
        public KeypadType DigitsTransmissionKeypadType;
        public DigitPosition DigitsPos;
        public KeyboardLayout KbdLayout;
        public HIDCharTransmitMode HidCharTransmitMode;
}
```

Members

	Name	Description
public	KbdType	Get or set the Bluetooth® HID keyboard interface type
public	InterCharDelay	Get or set the inter-character delay of the HID keyboard interface
public	LockState	Get or set the Caps Lock state of the HID keyboard interface
public	IgnoreCaseOnAlphabetsTransmission	Get or set a value indicating whether the case will be ignored during the transmission of alphabets
public	DigitsTransmissionKeypadType	Get or set the digits transmission type of the HID keyboard interface
public	DigitsPos	Get or set the digits position of the HID keyboard interface
public	KbdLayout	Get or set the keyboard layout of the HID keyboard interface
public	HidCharTransmitMode	Get or set the character transmission mode of the HID keyboard interface

See Also

2.5 CapsLockState
2.10 DigitPosition
2.15 HIDCharTransmitMode
2.16 HIDKeyboardType
2.19 KeyboardLayout
2.20 KeypadType
Configurator.GetConnectionInterface3610Hid Method
Configurator.GetConnectionInterfaceHid Method
Configurator.SetConnectionInterface3610Hid Method
Configurator.SetConnectionInterfaceHid Method

3.5 MemoryData

The **MemoryData** structure defines the record stored in the device memory.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public struct MemoryData
{
        public struct CounterSection
        {
                Public byte[] Prefix;
                Public byte[] Suffix;
                Public int Value;
        }

        public struct TimestampSection
        {
                public byte[] Prefix;
                public byte[] Suffix;
                public DateTime Value;
        }
        public struct DataSection
        {
                public byte[] Prefix;
                public byte[] Suffix;
                public string PC;
                public string EPC;
                public string CRC;
                public int Length;
        }

        public CounterSection Counter;
        public TimestampSection Timestamp;
        public DataSection Data;
        public String AsciiString;
}
```

Members

	Name	Description
public	Counter	CounterSection structure that contains tag counter prefix, suffix and value.
public	Timestamp	TimestampSection structure that contains timestamp prefix, suffix and value.
public	Data	DataSection structure that contains tag data prefix, suffix, EPC, PC, CRC, and data length.
public	AsciiString	String representation of data record

See Also

2.7 DataSequenceSection
Configurator.ReadDeviceMemoryData Method

3.6 RFIDData

The **RFIDData** structure defines the output data format of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public struct RFIDData
{
        public RFIDDataField[] Field;
        public char[] Separator;
        public RFIDData(int size)
        {
                Field = new RFIDDataField[size];
                Separator = new char[size - 1];
        }
}
```

Constructor

	Name	Description
public	RFIDData (int size)	Initializes a new instance of the RFIDData struct with the number of data fields.

Members

	Name	Description
public	Field	Data field
public	Separator	Separator between data fields

See Also

2.25 RFIDDataField
GetRFIDDataFormat Method (byte[], byte[], RFIDData)
GetRFIDDataFormat Method (string, string, RFIDData)
Configurator.SetRFIDDataFormat Method (byte[], byte[], RFIDData)
Configurator.SetRFIDDataFormat Method (string, string, RFIDData)

3.7 KeyActionString

The **KeyActionString** structure contains the actions/events of the 1800 device hardware keys and the corresponding strings that will be send when the key action is triggered.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public struct KeyActionString
{
      public string text;
       public KeyAction action;
}
```

Members

	Name	Description
public	Text	String to send when the defined key action is triggered
public	Action	Key action/event

See Also

2.18 KeyAction
Configurator.GetKeyActionString Method
Configurator.SetKeyActionString Method

3.8 Timestamp

The **Timestamp** structure defines the timestamp format of the output data.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public struct Timestamp
{
        public TimestampField[] Field;
        public char[] Separator;
        public YearFormat FormatOfYear;
        public Timestamp(int size)
        {
                Field = new TimestampField[size];
                Separator = new char[size - 1];
                ShowMilliseconds = true;
                FormatOfYear = YearFormat.FourDigits;
        }
}
```

Constructor

	Name	Description
public	Timestamp (int size)	Initializes a new instance of the Timestamp struct with the number of timestamp fields.

Members

	Name	Description
public	Field	Field of timestamp in output data
public	Separator	Separator between timestamp fields
public	FormatOfYear	2-digit or 4 digit year format

See Also

2.7 DataSequenceSection
2.30 TimestampField
Configurator.GetTimestampFormat Method (byte[], byte[], Timestamp)
Configurator.GetTimestampFormat Method (string, string, Timestamp)
Configurator.SetTimestampFormat Method (byte[], byte[], Timestamp)
Configurator.SetTimestampFormat Method (string, string, Timestamp)

Chapter 4

CLASSES

The **CipherLab.RFID.Device1800.Config** namespace contains classes that provide services enabling *Bluetooth*® or USB connection, configuration for 1800 Series Device, *Bluetooth*® device management, and supporting output data manipulation.

In This Chapter

4.1 BluetoothSocketConnector

Represents the Bluetooth Socket Connector.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public class BluetoothSocketConnector : Connector
```

Constructor

	Name	Description
public	BluetoothSocketConnector()	Initializes a new instance of the **BluetoothSocketConnector** class.

Properties

	Name	Description
public	Device	Gets or sets the target BTDevice to connect to.
public	PinCode	Gets or sets the Bluetooth® pin code for connecting to the target BTDevice.

Methods

	Name	Description
public	Connect()	Connects to the given BTDevice.
public	Disconnect()	Disconnects from the given BTDevice.
public	DiscoverDevice()	Searches for 1800 Series devices.
public	GetPairedDevice()	Gets a list of paired 1800 Series devices. (Window Mobile only)
public	IsConnected()	Gets the connection status.

4.1.1 BluetoothSocketConnector Constructor

Initializes a new instance of the BluetoothSocketConnector.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public BluetoothSocketConnector()
```

4.1.2 BluetoothSocketConnector Property

BluetoothSocketConnector.Device Property

Gets or sets the target device BTDevice to connect to.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public BTDevice Device { get; set; }
```

BluetoothSocketConnector.PinCode Property

Gets or sets the *Bluetooth*® pin code for connecting to the target BTDevice.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public string PinCode { get; set; }
```

4.1.3 BluetoothSocketConnector Methods

BluetoothSocketConnector.Connect Method

Connects to the given BTDevice.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public override bool Connect()
```

Return Value
True if successfully connected. The method returns false otherwise.

BluetoothSocketConnector.Disconnect Method

Disconnects from the given BTDevice.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public override void Disconnect()
```

BluetoothSocketConnector.DiscoverDevice Method

Searches for 1800 Series devices.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public BTDevice[] DiscoverDevice()
```

Return Value
A list of 1800 Series devices. The method returns null if no device was found.

BluetoothSocketConnector.GetPairedDevice Method

Gets a list of paired 1800 Series devices. (Window Mobile only)

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public BTDevice[] GetPairedDevice()
```

Return Value
A list of paired 1800 Series devices. The method returns null if no device was found.

BluetoothSocketConnector.IsConnected Method

Gets the current connection status.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public override bool IsConnected()
```

Return Value
If the connector is connected, this method returns true.
This method returns false if the connector is disconnected.

4.2 BluetoothVCPConnector

Represents the Bluetooth Virtual COM Port Connector.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public class BluetoothVCPConnector : Connector
```

Constructor

	Name	Description
public	BluetoothVCPConnector(string)	Initializes a new instance of the **BluetoothVCPConnector** class with the given Bluetooth virtual COM port.

Properties

	Name	Description
public	PortName	Gets the name of the virtual COM port.

Methods

	Name	Description
public	Connect()	Connects to the given Bluetooth virtual COM port.
public	Disconnect()	Disconnects from the given Bluetooth virtual COM port.
public	IsConnected()	Gets the connection status.

4.2.1 BluetoothVCPConnector Constructor

Initializes a new instance of the BluetoothVCPConnector.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public BluetoothVCPConnector(string portName)
```

Parameter	Description
portName	Type: string The name of the virtual COM port to connect to (e.g. COM17).

4.2.2 BluetoothVCPConnector Property

BluetoothVCPConnector.PortName Property

Gets the name of the virtual COM port.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public string PortName { get; }
```

4.2.3 BluetoothVCPConnector Methods

BluetoothVCPConnector.Connect Method

Connects to the given Bluetooth virtual COM port.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public override bool Connect()
```

Return Value
True if successfully connected. The method returns false otherwise.

BluetoothVCPConnector.Disconnect Method

Disconnects from the given Bluetooth virtual COM port.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public override void Disconnect()
```

BluetoothVCPConnector.IsConnected Method

Gets the current connection status.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public override bool IsConnected()
```

Return Value
If the connector is connected, this method returns true.
This method returns false if the connector is disconnected.

4.3 SerialPortConnector

Represents the Serial Port Connector.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public class SerialPortConnector : Connector
```

Constructor

	Name	Description
public	SerialPortConnector()	Initializes a new instance of the SerialPortConnector.

Property

	Name	Description
public	PortName	Gets the connected port name.

Methods

	Name	Description
public	Connect()	Connects to the first found serial port that uses USB virtual COM CDC driver.
public	Disconnect()	Disconnects from the 1800 Series device and close the serial port it used.
public	FindDeviceSerialPort()	Finds the first serial port that uses USB virtual COM CDC driver.
public	IsConnected()	Gets the connection status.

4.3.1 SerialPortConnector Constructor

Initializes a new instance of the SerialPortConnector.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public SerialPortConnector()
```

4.3.2 SerialPortConnector Methods

SerialConnector.Connect Method

Connects to the first found serial port that uses USB virtual COM CDC driver.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public override bool Connect()
```

Return Value
True if successfully connected. The method returns false otherwise.

SerialPortConnector.Disconnect Method

Disconnects from the 1800 Series device and close the serial port it used.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public override void Disconnect()
```

SerialPortConnector.IsConnected Method

Gets the current connection status.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public override bool IsConnected()
```

Return Value
If the connector is connected, this method returns true.
This method returns false if the connector is disconnected.

4.4 Configurator

Represents the 1800 Series device configurator.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public class Configurator
```

Constructor

	Name	Description
public	Configurator(Connector)	Initializes a new instance of the Configurator with a given Connector that is already connected.

Property

	Name	Description
public	Version	Gets the assembly version number.

Methods – I

	Name	Description
public	ClearDeviceMemoryData	Clears the 1800 device internal memory.
public	DisconnectBluetooth	Disconnects the 1800 device's *Bluetooth*® SPP Master connection and resets the device to *Bluetooth*® SPP Slave after.
public	EnableBluetoothPowerSaving	Enables or disables *Bluetooth*® power saving.
public	EnableBluetoothSecureSimplePairing	Enables or disables *Bluetooth*® Secure Simple Pairing.
public	EnableCommandBeep	Enables or disables the 1800 device command beeper.
public	EnableDeviceTrigger	Enables or disables the 1800 device's scan trigger key.
public	GetBatteryLifePercent	Gets the 1800 device battery life in percentage.
public	GetBluetoothInterfaceType	Gets the current *Bluetooth*® Interface.
public	GetConnectionInterface3610Hid	Gets the 3610 HID connection settings of the 1800 device.
public	GetConnectionInterface3610Vcom	Gets the serial number of the CipherLab 3610 dongle which the 1800 device is connected to.
public	GetConnectionInterfaceHid	Gets the *Bluetooth*® HID connection settings of the 1800 device.

Methods – II

	Name	Description
public	*GetConncetionInterfaceSppMaster*	Gets *Bluetooth*® SPP Master connection settings of the 1800 device.
public	*GetConnectionInterfaceSppSlave*	Gets *Bluetooth*® SPP Slave connection settings of the 1800 device.
public	*GetCounterFormat (byte[], byte[], char)*	Gets counter format settings.
public	*GetCounterFormat (string, string, char)*	Gets counter format settings.
public	*GetCounterResetOptions*	Gets the 1800 device counter reset settings.
public	*GetDataPortal*	Gets the the 1800 device's data outlet.
public	*GetDeviceInfo*	Gets the firmware information about the 1800 device.
public	*GetDeviceNotificationState*	Gets the notification settings of the 1800 device, including system and good read beeper, low battery alert, LED and vibrator.
public	*GetDeviceOperatingMode*	Gets the 1800 device operating mode.
public	*GetDevicePowerSavingState*	Gets the power saving timeouts of the 1800 device.
public	*GetDeviceTime*	Gets the 1800 device clock date and time.
public	*GetEPCFilter*	Gets the EPC filter settings of the 1800 device.
public	*GetKeyActionString*	Gets the 1800 device hardware keys actions and the corresponding strings that will be sent when the key action is triggered.
public	*GetMemoryModeDelay*	Gets the delay time of memory mode data transmission.
public	*GetMultiTagListType*	Gets the data type in which the multi-tag list of the 1800 device should store in order to avoid duplicate tag scans.
public	*GetMultiTagModeCounterLimit*	Gets the counter limit of the current multi-tag settings.
public	*GetMultiTagModeSettings*	Gets the multi-tag scan mode settings of the 1800 device.
public	*GetOnlineModeDelay*	Gets the delay time of online mode data transmission.
public	*GetOutputDataFormat*	Gets the output data format of the 1800 device.

Methods – III

	Name	Description
public	*GetOutputDataSequence*	Gets the sequence of output data.
public	*GetQValue*	Gets the Q value.
Public	*GetRecognizedEPCEncoding*	Gets the EPC encoding schemes that are recognized by the 1800 device.
public	*GetRFIDDataFormat (byte[], byte[], RFIDData)*	Gets RFID data format settings.
public	*GetRFIDDataFormat (string, string, RFIDData)*	Gets RFID data format settings.
public	*GetRFIDMode*	Gets the RFID scan mode of the 1800 device.
public	*GetRFIDPowerLevel*	Gets the 1800 device's RFID module signal strength level from 0 to 3. Bigger value means stronger signal strength. By default, the value is set to 3.
public	*GetRFIDSwitchStatus*	Gets the 1800 device's reader switch position – RFID or EXT. mode.
public	*GetScanMode*	Gets the current scan mode.
public	*GetSelectedMemoryBank*	Gets the current selected memory bank.
public	*GetSingleTagModeSessionTimeout*	Gets the current scan session timeout in single tag scan mode.
public	*GetTimestampFormat (byte[], byte[], Timestamp)*	Gets the format of timestamp settings.
public	*GetTimestampFormat (string, string, Timestamp)*	Gets the format of timestamp settings.
public	*GetUSBDriver*	Gets the current USB driver in use.
public	*IsBluetoothPowerSavingEnabled*	Gets *Bluetooth*® power saving status.
public	*IsBluetoothSecureSimplePairing*	Gets the value indicating whether *Bluetooth*® Secure Simple Pairing is enabled.
public	*IsBT3610SetToVCOM*	Gets a value indicating whether the 3610 connection interface is set to VCOM.
public	*KeepDeviceAlive*	Starts a background thread to prevent the 1800 device from entering power-saving mode.
public	*LoadFactoryDefaultSettings*	Resets the 1800 device to its factory default settings.
public	*LoadUserSettings*	Resets the 1800 device with user setting.
public	*ReadDeviceMemoryData*	Reads the device memory data.
public	*ResetDataCounter*	Resets the tag counter.

Methods – IV

	Name	Description
public	*RFIDDirectCancelInventoryRound*	Stops the 1800 device reading while it is in the middle of an inventory round.
public	*RFIDDirectKillTag*	Triggers the 1800 device to kill a RFID tag with a given EPC.
public	*RFIDDirectLockTag*	Triggers the 1800 device to lock a RFID tag memory bank with a given EPC.
public	*RFIDDirectReadTagByEPC*	Triggers a RFID tag read on the 1800 device with a given EPC.
public	*RFIDDirectReadTagByTID*	Triggers a RFID tag read on the 1800 device with a given TID.
public	*RFIDDirectPermanentLockTag*	Triggers the 1800 device to permanently lock a RFID tag memory bank with a given EPC.
public	*RFIDDirectStartInventoryRound*	Triggers a RFID inventory round on the 1800 device.
public	*RFIDDirectUnlockTag*	Triggers the 1800 device to unlock a RFID tag memory bank with a given EPC.
public	*RFIDDirectWriteTagByEPC*	Triggers a RFID tag write on the 1800 device with a given EPC.
public	*RFIDDirectWriteTagByTID*	Triggers a RFID tag write on the 1800 device with a given TID.
public	*RFIDReadTagMassive*	Prepares the 1800 device for trigger-pressed read.
public	*RFIDWriteTagEPCBank*	Prepares the 1800 device for trigger-pressed EPC bank write.
public	*RFIDWriteTagMassive*	Prepares the 1800 device for trigger-pressed write.
public	*RFIDWriteTagPassword*	Prepares the 1800 device for trigger-pressed reserved bank write
public	*SaveUserSettings*	Saves the current 1800 device setting as the user setting.
public	*SelectMemoryBank*	Selects a memory bank to read or write.
public	*SetBluetoothInterfaceType*	Changes the current *Bluetooth*® Interface.
public	*SetConnectionInterface3610Hid*	Configures the 3610 HID connection settings of the 1800 device.
Public	*SetConnectionInterface3610Vcom*	Sets the serial number of the CipherLab 3610 dongle which you'd like to connect your 1800 device to.

Methods – V

	Name	Description
public	*SetConnectionInterfaceHid*	Configures the *Bluetooth*® HID connection settings of the 1800 device.
public	*SetConnectionInterfaceSppMaster*	Configures the *Bluetooth*® SPP Master connection settings of the 1800 device.
public	*SetConnectionInterfaceSppSlave*	Configures the *Bluetooth*® SPP Slave connection settings of the 1800 device.
public	*SetCounterFormat (byte[], byte[], char)*	Sets counter format.
public	*SetCounterFormat (string, string, char)*	Sets counter format.
public	*SetCounterResetOptions*	Sets the times to reset the counter on the 1800 device.
public	*SetDataPortal*	Sets the 1800 device's data outlet.
public	*SetDeviceNotificationState*	Changes the notification settings of the 1800 device, including system and good read beep, low battery alert, LED indication and vibration.
public	*SetDeviceOperatingMode*	Changes the 1800 device operating mode.
public	*SetDevicePowerSavingState*	Sets the power saving timeouts of the 1800 device. By default, the idle power saving timeout is set to 2 minutes and the idle shutdown timeout is set to 10 minutes.
public	*SetDeviceTime*	Changes the 1800 device clock. Year format can be specifed as four-digit year (e.g. 2012) or two-digit year(e.g. 12).
public	*SetEPCFilter*	Sets how the 1800 device filters RFID tags.
public	*SetFirmwareUpdateInterface*	Sets firmware update interface of the 1800 device.
public	*SetKeyActionString*	Sets the 1800 device hardware keys actions and the corresponding strings that will be sent when the key action is triggered.
public	*SetMemoryModeDelay*	Sets the delay time of Memory mode data transmission.
public	*SetMultiTagListType*	Sets the data type in which the multi-tag list of the 1800 device should store in order to avoid duplicate tag scans
public	*SetMultiTagModeCounterLimit*	Changes the counter limit of current multi-tag settings.

Methods – VI

	Name	Description
public	*SetMultiTagModeSettings*	Changes the multi-tag settings of the 1800 device.
public	*SetOnlineModeDelay*	Changes the delay time of online mode data transmission.
public	*SetOutputDataFormat*	Sets the data output data format of the 1800 device.
public	*SetOutputDataSequence*	Sets the sequence of output data.
public	*SetQValue*	Sets the Q value.
public	*SetRecognizedEPCEncoding*	Sets the EPC encoding schemes that are recognized by the 1800 device.
public	*SetRFIDMode*	Sets the RFID scan mode of the 1800 device.
public	*SetRFIDDataFormat (byte[], byte[], RFID)*	Sets RFID data format.
public	*SetRFIDDataFormat (string, string, RFID)*	Sets RFID data format.
public	*SetRFIDPowerLevel*	Sets the 1800 device RFID module signal strength level.
public	*SetScanMode*	Changes the current scan mode.
public	*SetSingleTagModeSessionTimeout*	Sets current scan session timeout in single tag scan mode.
public	*SetTimestampFormat (byte[], byte[], Timestamp)*	Sets the format of timestamp.
public	*SetTimestampFormat (string, string, Timestamp)*	Sets the format of timestamp.
public	*SetUSBDriver*	Sets the current USB driver in use.
public	*ShutdownDevice*	Shuts down the 1800 device.

4.4.1 Configurator Constructor (Connector)

Initializes a new instance of the Configurator with a given Connector that is already connected.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public Configurator(Connector connector)
```

Remarks
Connector must be connected prior to the instantiation of the Configurator object.

4.4.2 Configurator Property

Configurator.Version Property

Gets the assembly version number.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public string Version { get; }
```

4.4.3 Configurator Methods

Configurator.ClearDeviceMemoryData Method

Clears the 1800 device internal memory.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void ClearDeviceMemoryData()
```

Example

```
using CipherLab.RFID.Device1800.Config;

config.ClearDeviceMemeoryData();
```

See Also

Configurator.ReadDeviceMemoryData Method

Back to List of Configurator Methods - I

Configurator.DisconnectBluetooth Method

Disconnects the 1800 device's *Bluetooth*® SPP Master connection and clears the connection record (Bluetooth address of the last connected device). It also resets the device to *Bluetooth*® SPP Slave after if told.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void DisconnectBluetooth (
        bool reset2SppSlave
)
```

Parameter	Description
reset2SppSlave	Type: bool Resets the 1800 device to Bluetooth SPP Slave after its Bluetooth connection disconnects.

Remarks

This method is only available in the Windows version of the SDK.
It will cause the BT module of the 1800 device to re-initialize, which takes approximately 3.6 seconds.

Example

```
using CipherLab.RFID.Device1800.Config;

config.DisconnectBluetooth(true);
```

See Also

Configurator.GetConnectionInterfaceSppMaster Method
Configurator.SetConnectionInterfaceSppMaster Method

Back to List of Configurator Methods - I

Configurator.EnableBluetoothPowerSaving Method

Enables or disables *Bluetooth*® power saving.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void EnableBluetoothPowerSaving(
        bool enabled
)
```

Parameter	Description
enabled	Type: bool "True" to enable Bluetooth® power saving. When enabled, the 1800 device will listen to the wireless network at a reduced rate. "False" to disable Bluetooth® power saving.

Example

```
using CipherLab.RFID.Device1800.Config;

config.EnableBluetoothPowerSaving(true);
```

See Also

Configurator.IsBluetoothPowerSavingEnabled Method

Back to List of Configurator Methods - I

Configurator.EnableBluetoothSecureSimplePairing Method

Enables or disables *Bluetooth*® Secure Simple Pairing.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void EnableBluetoothSecureSimplePairing(
        bool enabled
)
```

Parameter	Description
enabled	Type: bool "True" to enable Bluetooth® Secure Simple Pairing. When enabled, the 1800 device will auto determine the best pairing method. "False" to disable Bluetooth® Secure Simple Pairing.

Example

```
using CipherLab.RFID.Device1800.Config;

config.EnableBluetoothSecureSimplePairing(true);
```

See Also

Configurator.IsBluetoothSecureSimplePairingEnabled Method

Back to List of Configurator Methods - I

Configurator.EnableCommandBeep Method

Enables or disables the 1800 device command beeper.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void EnableCommandBeep(
        bool enabled
)
```

Parameter	Description
enabled	Type: bool "True" to enable the command beeper; "False" to disable the command beeper.

Example

```
using CipherLab.RFID.Device1800.Config;

config.EnableCommandBeep(false);
```

Back to List of Configurator Methods - I

Configurator.EnableDeviceTrigger Method

Enables or disables the 1800 device's scan trigger key.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void EnableDeviceTrigger(
        bool enabled)
```

Return Value
Type: bool
Scan trigger key is enabled or disabled.

Example

```
using CipherLab.RFID.Device1800.Config;

config.EnableDeviceTrigger(false);
```

Back to List of Configurator Methods - I

Configurator.GetBatteryLifePercent Method

Gets the 1800 device battery life in percentage.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public int GetBatteryLifePercent()
```

Return Value
Type: int
Percentages values: 5, 20, 40, 60, 80, and 100.

Example

```
using CipherLab.RFID.Device1800.Config;

int Battery = config.GetBatteryLifePercent();
string str = String.Format("Battery = {0}%", Battery);
```

Back to List of Configurator Methods - I

Configurator.GetBluetoothInterfaceType Method

Gets the current *Bluetooth*® Interface.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public BluetoothInterface GetBluetoothInterfaceType()
```

Return Value

Type: BluetoothInterface
See **BluetoothInterface** enumeration for a list of Bluetooth® connection interface supported by the 1800 device.

Example

```
using CipherLab.RFID.Device1800.Config;

BluetoothInterface Interface_type =
config.GetBluetoothInterfaceType();
```

See Also

2.4 BluetoothInterface
Configurator.SetBluetoothInterfaceType Method

Back to List of Configurator Methods - I

Configurator.GetConnectionInterface3610Hid Method

Gets the 3610 HID connection settings of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void GetConnectionInterface3610Hid(
        out HID hid,
        out string serialNumber
)
```

Parameter	Description
hid	Type: HID 3610 HID settings
serialNumber	Type: string The serial number of CipherLab 3610 dongle has 9 characters.

Remarks

This method is only available in the Windows version of the SDK.

Example

```
using CipherLab.RFID.Device1800.Config;

string serialNumber = "";
HID hid = new HID();
config. GetConnectionInterface3610Hid(out serialNumber, out hid);
```

See Also

3.4 HID Configurator.SetConnectionInterface3610Hid Method

Back to List of Configurator Methods - I

Configurator.GetConnectionInterface3610Vcom Method

Gets the serial number of the CipherLab 3610 dongle which the 1800 device is connected to.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public bool GetConnectionInterface3610Vcom(
        out string serialNumber
)
```

Parameter	Description
serialNumber	Type: string The serial number f of CipherLab 3610 dongle has 9 characters.

Return Value

Type: bool

Remarks

This method is only available in the Windows version of the SDK.

Example

```
using CipherLab.RFID.Device1800.Config;

string serialNumber = "";
bool r = config.GetConnectionInterface3610Vcom(serialNumber);
```

See Also

Configurator.SetConnectionInterface3610Vcom Method

Back to List of Configurator Methods - I

Configurator.GetConnectionInterfaceHid Method

Gets the *Bluetooth*® HID connection settings of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void GetConnectionInterfaceHid(
        out bool visible,
        out bool secured,
        out string pin,
        out HID hid
)
```

Parameter	Description
visible	Type: bool A value indicating whether the 1800 device can be discovered by other Bluetooth® devices
secured	Type: bool A value indicating whether Bluetooth® authentication is enabled for the 1800 device.
pin	Type: string Bluetooth® PIN for pairing: 1~16 digits **Default PIN=0000**
hid	Type: HID Bluetooth® HID settings

Example

```
using CipherLab.RFID.Device1800.Config;

bool visible = false;
bool secured=false;
string pin="";
HID hid = new HID();

config.GetConnectionInterfaceHid(out visible, out secured, out pin, out hid);
```

See Also

3.4 HID Configurator.SetConnectionInterfaceHid Method

Back to List of Configurator Methods - I

Configurator.GetConnectionInterfaceSppMaster Method

Gets *Bluetooth*® SPP Master connection settings of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public bool GetConncetionInterfaceSppMaster(
        out string btAddr,
        out bool visible,
        out bool secured,
        out string pin
)
```

Parameter	Description
btAddr	Type: string Hexadecimal representation of the Bluetooth® address of device or computer to connect to without separators.
visible	Type: bool A value indicating whether the 1800 device can be discovered by other Bluetooth® devices.
secured	Type: bool A value indicating whether Bluetooth® authentication is enabled for the 1800 device.
pin	Type: string Bluetooth® PIN for pairing: 1~16 digits **Default PIN=0000**

Return Value

Type: bool

Remarks

This method is only available in the Windows version of the SDK.
Bluetooth address will not be overwritten until a successful connection has been established.

Example

```
using CipherLab.RFID.Device1800.Config;

string btAddr;
string pin = "";
bool visible = false;
bool secured = false;
config.GetConncetionInterfaceSppMaster(out btAddr, out visible, out
secured, out pin);
```

See Also

Configurator.DisconnectBluetooth Method
Configurator.SetConnectionInterfaceSppMaster Method

Back to List of Configurator Methods - II

Configurator.GetConnectionInterfaceSppSlave Method

Gets *Bluetooth*® SPP Slave connection settings of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public bool GetConncetionInterfaceSppSlave(
        out bool visible,
        out bool secured,
        out string pin
)
```

Parameter	Description
visible	Type: bool A value indicating whether the 1800 device can be discovered by other Bluetooth® devices.
secured	Type: bool A value indicating whether Bluetooth® authentication is enabled for the 1800 device.
pin	Type: string Bluetooth® PIN for pairing: 1~16 digits **Default PIN=0000**

Return Value

Type: bool

Example

```
using CipherLab.RFID.Device1800.Config;

string pin = "";
bool visible = false;
bool secured = false;
config.GetConncetionInterfaceSppSlave(out visible, out secured, out
pin);
```

See Also

Configurator.SetConnectionInterfaceSppSlave Method

Back to List of Configurator Methods - II

Configurator.GetCounterFormat Method (byte[], byte[], char)

Gets counter format settings.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void GetCounterFormat(
        out byte[] prefix,
        out byte[] suffix,
        out char leftPaddingChar
)
```

Parameter	Description
prefix	Type: byte[] Counter prefix in bytes. Max. 8 bytes allowed. Null to disable counter prefix.
suffix	Type: byte[] Counter suffix in bytes. Max. 8 bytes allowed. Null to disable counter suffix.
leftPaddingChar	Type: char A character to pad before counter.

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] prefix;
byte[] suffix;
char leftPaddingChar = '0';
config.GetCounterFormat(out prefix, out suffix, out leftPaddingChar);
```

See Also

2.7 DataSequenceSection
Configurator.GetCounterFormat Method (string, string, char)
Configurator.SetCounterFormat Method (byte[], byte[], char)
Configurator.SetCounterFormat Method (string, string, char)

Back to List of Configurator Methods - II

Configurator.GetCounterFormat Method (string, string, char)

Gets counter format settings.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void GetCounterFormat(
        out string prefix,
        out string suffix,
        out char leftPaddingChar
)
```

Parameter	Description
prefix	Type: string Counter prefix in string format. Max. 8 characters allowed.
suffix	Type: string Counter suffix in string format. Max. 8 characters allowed.
leftPaddingChar	Type: char A character to pad before counter.

Example

```
using CipherLab.RFID.Device1800.Config;

string prefix = "";
string suffix = "";
char leftPaddingChar = '0';
config.GetCounterFormat(out prefix, out suffix, out leftPaddingChar);
```

See Also

2.7 DataSequenceSection
Configurator.GetCounterFormat Method (byte[], byte[], char)
Configurator.SetCounterFormat Method (byte[], byte[], char)
Configurator.SetCounterFormat Method (string, string, char)

Back to List of Configurator Methods - II

Configurator.GetCounterResetOptions Method

Gets the 1800 device counter reset settings.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void GetCounterResetOptions(
        out bool byCommand,
        out bool rfidPowerOn,
        out bool newBTConnection
)
```

Parameter	Description
byCommand	Type: bool A value indicating whether the counter is reset when a reset command is received.
rfidPowerOn	Type: bool A value indicating whether the counter is reset when the RFID module powers on.
newBTConnection	Type: bool A value indicating whether the counter is reset when a new Bluetooth® connection is established

Example

```
using CipherLab.RFID.Device1800.Config;

bool byCommand = false;
bool rfidPowerOn = false;
bool newBTConnection = false;
config.GetCounterResetOptions(out byCommand, out rfidPowerOn, out
newBTConnection);
```

See Also

Configurator.GetMultiTagModeCounterLimit Method
Configurator.GetMultiTagModeSettings Method
Configurator.ResetDataCounter Method
Configurator.SetCounterResetOptions Method
Configurator.SetMultiTagModeCounterLimit Method
Configurator.SetMultiTagModeSettings Method

Back to List of Configurator Methods - II

Configurator.GetDataPortal Method

Gets the the 1800 device's data outlet.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public DataPortal GetDataPortal()
```

Return Value
Type: DataPortal

Example

```
using CipherLab.RFID.Device1800.Config;

DataPortal outlet = config.GetDataPortal();
```

See Also
2.6 DataPortal
Configurator.SetDataPortal Method

Back to List of Configurator Methods - II

Configurator.GetDeviceInfo Method

Gets the firmware information about the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public DeviceInfo GetDeviceInfo()
```

Return Value

Type: DeviceInfo

Example

```
using CipherLab.RFID.Device1800.Config;

DeviceInfo info = config.GetDeviceInfo();
lbl_model_value.Text = info.Model;
lbl_serial_value.Text = info.SerialNumber;
lbl_firmware_value.Text = info.UserVersion;
lbl_bt_value.Text = info.MacAddress;
```

See Also

3.1 DeviceInfo

Back to List of Configurator Methods - II

Configurator.GetDeviceNotificationState Method

Gets the notification settings of the 1800 device, including system and good read beeper, low battery alert, LED and vibrator.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public DeviceNotification GetDeviceNotificationState()
```

Return Value

Type: DeviceNotification

Example

```
using CipherLab.RFID.Device1800.Config;

DeviceNotification norification = config.GetDeviceNotificationState();
```

See Also

3.2 DeviceNotification
Configurator.SetDeviceNotificationState Method

Back to List of Configurator Methods - II

Configurator.GetDeviceOperatingMode Method

Gets the 1800 device operating mode.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public OperatingMode GetDeviceOperatingMode()
```

Return Value

Type: OperatingMode

Example

```
using CipherLab.RFID.Device1800.Config;

OperatingMode System_MemoryMode = config.GetDeviceOperatingMode();
```

See Also

2.23 OperatingMode
Configurator.SetDeviceOperatingMode Method

Back to List of Configurator Methods - II

Configurator.GetDevicePowerSavingState Method

Gets the power saving timeouts of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void GetDevicePowerSavingState(
        out int standby,
        out int shutdown
)
```

Parameter	Description
standby	Type: int Idle minutes to enter power saving mode. Maximum value is 254 minutes. If set to 0, the 1800 device never enters power saving mode.
shutdown	Type: int Idle minutes to shut down the device. Maximum value is 254 minutes. If set to 0, the 1800 idle shutdown is disabled.

Example

```
using CipherLab.RFID.Device1800.Config;

int PowerSavingDelayTime = 0;
int ShutdownDelayTime = 0;
config.GetDevicePowerSavingState(out PowerSavingDelayTime,
out ShutdownDelayTime);
```

See Also

Configurator.SetDevicePowerSavingState Method

Back to List of Configurator Methods - II

Configurator.GetDeviceTime Method

Gets the 1800 device clock date and time.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public DateTime GetDeviceTime()
```

Return Value
Type: DateTime

Example

```
using CipherLab.RFID.Device1800.Config;

DateTime currect = config.GetDeviceTime();
```

See Also
Configurator.SetDeviceTime Method

Back to List of Configurator Methods - II

Configurator.GetEPCFilter Method

Gets the EPC filter settings of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public EPCFilter GetEPCFilter()
```

Return Value

Type: EPCFilter
See **EPCFilter** structure.

Example

```
using CipherLab.RFID.Device1800.Config;

EPCFilter filter = config.GetEPCFilter();
```

See Also

2.12 EPCFilterType
2.13 FilterMethod
3.3 EPCFilter
Configurator.SetEPCFilter Method

Back to List of Configurator Methods - II

Configurator.GetKeyActionString Method

Gets the 1800 device hardware keys actions and the corresponding strings that will be sent when the key action is triggered.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public KeyActionString[] GetKeyActionString()
```

Return Value

Type: KeyActionString[]
See **KeyActionString** structure.

Example

```
using CipherLab.RFID.Device1800.Config;

KeyActionString[] OUTPUT_Action = config.GetKeyActionString();
```

See Also

2.18 KeyAction
3.7 KeyActionString
Configurator.SetKeyActionString Method

Back to List of Configurator Methods - II

Configurator.GetMemoryModeDelay Method

Gets the delay time of memory mode data transmission.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public TransmissionDelay GetMemoryModeDelay()
```

Return Value

Type: TransmissionDelay
See **TransmissionDelay** enumeration for a list of delay values.

Example

```
using CipherLab.RFID.Device1800.Config;

TransmissionDelay System_DataTransmissionDelay =
config.GetMemoryModeDelay();
```

See Also

Configurator.SetDeviceOperatingMode Method
Configurator.SetMemoryModeDelay Method

Back to List of Configurator Methods - II

Configurator.GetMultiTagListType Method

Gets the data type in which the multi-tag list of the 1800 device should store in order to avoid duplicate tag scans.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public MultiTagListType GetMultiTagListType()
```

Return Value

Type: MultiTagListType

Example

```
using CipherLab.RFID.Device1800.Config;

MultiTagListType type = config.GetMultiTagListType();
```

See Also

2.22 MultiTagListType
Configurator.SetMultiTagListType Method

Back to List of Configurator Methods - II

Configurator.GetMultiTagModeCounterLimit Method

Gets the counter limit of the current multi-tag settings.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public int GetMultiTagModeCounterLimit()
```

Return Value

Type: int

Example

```
using CipherLab.RFID.Device1800.Config;

int count = config.GetMultiTagModeCounterLimit();
```

See Also

Configurator.GetCounterResetOptions Method
Configurator.GetMultiTagModeSettings Method
Configurator.ResetDataCounter Method
Configurator.SetCounterResetOptions Method
Configurator.SetMultiTagModeCounterLimit Method
Configurator.SetMultiTagModeSettings Method

Back to List of Configurator Methods - II

Configurator.GetMultiTagModeSettings Method

Gets the multi-tag scan mode settings of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void GetMultiTagModeSettings(
        out ScanDelay scanDelay,
        out int counterLimit,
        out bool warnLimitReached
)
```

Parameter	Description
scanDelay	Type: ScanDelay See **ScanDelay** enumeration for a list of delay time intervals.
counterLimit	Type: int Limit of multi-tag mode counter for avoiding duplicate tag scans. The legal counter range is from 0 to 128. When the counter is set to 0, duplicate tag scan is allowed. Default value is 0.
warnLimitReached	Type: bool Beep if the multi-tag list is full.

Example

```
using CipherLab.RFID.Device1800.Config;

ScanDelay sd = ScanDelay.Delay_100ms;
int counterLimit = 0;
bool warnLimitReached = false;
config. GetMultiTagModeSettings (out sd, out counterLimit, out
warnLimitReached);
```

See Also

2.28 ScanDelay
Configurator.GetCounterResetOptions Method
Configurator.GetMultiTagModeCounterLimit Method
Configurator.ResetDataCounter Method
Configurator.SetCounterResetOptions Method
Configurator.SetMultiTagModeCounterLimit Method
Configurator.SetMultiTagModeSettings Method

Back to List of Configurator Methods - II

Configurator.GetOnlineModeDelay Method

Gets the delay time of online mode data transmission.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public TransmissionDelay GetOnlineModeDelay()
```

Return Value

Type: TransmissionDelay
See **TransmissionDelay** enumeration for a list of delay values.

Example

```
using CipherLab.RFID.Device1800.Config;

TransmissionDelay System_TransmissionDelay =
config.GetOnlineModeDelay();
```

See Also

2.31 TransmissionDelay
Configurator.SetOnlineModeDelay Method

Back to List of Configurator Methods - II

Configurator.GetOutputDataFormat Method

Gets the output data format of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public OutputDataFormat GetOutputDataFormat()
```

Return Value
Type: OutputDataFormat

Example

```
using CipherLab.RFID.Device1800.Config;

OutputDataFormat OUTPUT_Format = config.GetOutputDataFormat();
```

See Also
2.24 OutputDataFormat
Configurator.SetOutputDataFormat Method

Back to List of Configurator Methods - II

Configurator.GetOutputDataSequence Method

Gets the sequence of output data.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void GetOutputDataSequence(
        out DataSequenceSection section1,
        out DataSequenceSection section2,
        out DataSequenceSection section3
)
```

Parameter	Description
section1	Type: DataSequenceSection See **DataSequenceSection** enumeration.
section2	Type: DataSequenceSection See **DataSequenceSection** enumeration.
section3	Type: DataSequenceSection See **DataSequenceSection** enumeration.

Example

```
using CipherLab.RFID.Device1800.Config;

DataSequenceSection section1 = DataSequenceSection.NotUsed;
DataSequenceSection section2 = DataSequenceSection.NotUsed;
DataSequenceSection section3 = DataSequenceSection.NotUsed;
config.GetOutputDataSequence(out section1, out section2, out section3);
```

See Also

2.7 DataSequenceSection
Configurator.SetOutputDataSequence Method

Back to List of Configurator Methods - III

Configurator.GetQValue Method

Gets the Q value.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public int GetQValue()
```

Return Value

Type: int
The Q value.

Example

```
using CipherLab.RFID.Device1800.Config;

int value = config.GetQValue();
```

See Also

Configurator.SetQValue Method

Back to List of Configurator Methods - III

Configurator.GetRecognizedEPCEncoding Method

Gets the EPC encoding schemes that are recognized by the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public EPCEncodingScheme GetRecognizedEPCEncoding()
```

Return Value

Type: EPCEncodingScheme
See **EPCEncodingScheme** enumeration for a list of encoding schemes.

Example

```
using CipherLab.RFID.Device1800.Config;

EPCEncodingScheme encode = config.GetRecognizedEPCEncoding();
```

See Also

2.11 EPCEncodingScheme
Configurator.SetRecognizedEPCEncoding Method

Back to List of Configurator Methods - III

Configurator.GetRFIDDataFormat Method (byte[], byte[], RFIDData)

Gets RFID data format settings.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void GetRFIDDataFormat(
        out byte[] prefix,
        out byte[] suffix,
        out RFIDData dataFormat
)
```

Parameter	Description
prefix	Type: byte[] Data prefix in bytes. Max. 8 bytes allowed. Null to disable data prefix.
suffix	Type: byte[] Data suffix in bytes. Max. 8 bytes allowed. Null to disable data suffix.
dataFormat	Type: RFIDData See **RFIDData** structure.

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] prefix;
byte[] suffix;
RFIDData dataFormat = new RFIDData();
config.GetRFIDDataFormat(out prefix, out suffix, out dataFormat);
```

See Also

2.7 DataSequenceSection
2.25 RFIDDataField
3.6 RFIDData
Configurator.GetRFIDDataFormat Method (string, string, RFIDData)
Configurator.SetRFIDDataFormat Method (byte[], byte[], RFIDData)
Configurator.SetRFIDDataFormat Method (string, string, RFIDData)

Back to List of Configurator Methods - III

Configurator.GetRFIDDataFormat Method (string, string, RFIDData)

Gets RFID data format settings.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void GetRFIDDataFormat(
        out string prefix,
        out string suffix,
        out RFIDData dataFormat
)
```

Parameter	Description
prefix	Type: string Data prefix in string format. Max. 8 characters allowed.
suffix	Type: string Data suffix in string format. Max. 8 characters allowed.
dataFormat	Type: RFIDData See **RFIDData** structure.

Example

```
using CipherLab.RFID.Device1800.Config;

string prefix = "";
string suffix = "";
RFIDData dataFormat = new RFIDData();
config.GetRFIDDataFormat(out prefix, out suffix, out dataFormat);
```

See Also

2.7 DataSequenceSection
2.25 RFIDDataField
3.6 RFIDData
Configurator.GetRFIDDataFormat Method (byte[], byte[], RFIDData)
Configurator.SetRFIDDataFormat Method (byte[], byte[], RFIDData)
Configurator.SetRFIDDataFormat Method (string, string, RFIDData)

Back to List of Configurator Methods - III

Configurator.GetRFIDMode Method

Gets the RFID scan mode of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public RFIDMode GetRFIDMode()
```

Return Value
Type: RFIDMode
See **RFIDMode** enumeration.

Example

```
using CipherLab.RFID.Device1800.Config;

RFIDMode mode = config.GetRFIDMode();
```

See Also
2.27 RFIDMode
Configurator.SetRFIDMode Method

Back to List of Configurator Methods - III

Configurator.GetRFIDPowerLevel Method

Gets the 1800 device's RFID module signal strength level from 0 to 3. Bigger value means stronger signal strength. By default, the value is set to 3.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public int GetRFIDPowerLevel()
```

Return Value
Type: int

Example

```
using CipherLab.RFID.Device1800.Config;

int level = config.GetRFIDPowerLevel();
```

See Also
Configurator.SetRFIDPowerLevel Method

Back to List of Configurator Methods - III

Configurator.GetRFIDSwitchStatus Method

Gets the 1800 device's reader switch position – RFID or EXT. mode.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public bool GetRFIDSwitchStatus()
```

Return Value
Type: bool
True: Switch is set to RFID
False: Switch is set to EXT.

Example

```
using CipherLab.RFID.Device1800.Config;

bool result = config. GetRFIDSwitchStatus();
```

Back to List of Configurator Methods - III

Configurator.GetScanMode Method

Gets the current scan mode.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public ScanMode GetScanMode()
```

Return Value

Type: ScanMode
See **ScanMode** enumeration for a list of modes.

Example

```
using CipherLab.RFID.Device1800.Config;

ScanMode RFID_ScanMode = config.GetScanMode();
```

See Also

2.29 ScanMode
Configurator.SetScanMode Method

Back to List of Configurator Methods - III

Configurator.GetSelectedMemoryBank Method

Gets the current selected memory bank.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public RFIDMemoryBank GetSelectedMemoryBank()
```

Return Value

Type: RFIDMemoryBank

Example

```
using CipherLab.RFID.Device1800.Config;

RFIDMemoryBank bank = config.GetSelectedMemoryBank();
```

See Also

2.26 RFIDMemoryBank
Configurator.SelectMemoryBank Method

Back to List of Configurator Methods - III

Configurator.GetSingleTagModeSessionTimeout Method

Gets the current scan session timeout in single tag scan mode.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public int GetSingleTagModeSessionTimeout()
```

Return Value

Type: int

Example

```
using CipherLab.RFID.Device1800.Config;

int RFID_SingleTag_Timeout = config.GetSingleTagModeSessionTimeout();
```

See Also

Configurator.GetScanMode Method
Configurator.SetScanMode Method
Configurator.SetSingleTagModeSessionTimeout Method

Back to List of Configurator Methods - III

Configurator.GetTimestampFormat Method (byte[], byte[], Timestamp)

Gets the format of timestamp settings.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void GetTimestampFormat(
        out byte[] prefix,
        out byte[] suffix,
        out Timestamp timestampFormat
)
```

Parameter	Description
prefix	Type: byte[] Timestamp prefix in bytes. Max. 8 bytes allowed. Null to disable data prefix.
suffix	Type: byte[] Timestamp suffix in bytes. Max. 8 bytes allowed. Null to disable data suffix.
timestampFormat	Type: Timestamp See **Timestamp** structure.

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] prefix;
byte[] suffix;
Timestamp timestampFormat = new Timestamp();
config.GetTimestampFormat(out prefix, out suffix, out timestampFormat);
```

See Also

2.7 DataSequenceSection
2.30 TimestampField
3.8 Timestamp
Configurator.GetTimestampFormat Method (string, string, Timestamp)
Configurator.SetTimestampFormat Method (byte[], byte[], Timestamp)
Configurator.SetTimestampFormat Method (string, string, Timestamp)

Back to List of Configurator Methods - III

Configurator.GetTimestampFormat Method (string, string, Timestamp)

Gets time stamp format settings.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void GetTimestampFormat(
        out string prefix,
        out string suffix,
        out Timestamp timestampFormat
)
```

Parameter	Description
prefix	Type: string Timestamp prefix in string format. Max. 8 characters allowed.
suffix	Type: string Timestamp suffix in in string format. Max. 8 characters allowed.
timestampFormat	Type: Timestamp See **Timestamp** structure.

Example

```
using CipherLab.RFID.Device1800.Config;

string prefix = "";
string suffix = "";
Timestamp timestampFormat = new Timestamp();
config.GetTimestampFormat(out prefix, out suffix, out timestampFormat);
```

See Also

2.7 DataSequenceSection
2.30 TimestampField,
3.8 Timestamp
Configurator.GetTimestampFormat Method (byte[], byte[], Timestamp)
Configurator.SetTimestampFormat Method (byte[], byte[], Timestamp)
Configurator.SetTimestampFormat Method (string, string, Timestamp)

Back to List of Configurator Methods - III

Configurator.GetUSBDriver Method

Gets the current USB driver in use.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public USBDriver GetUSBDriver()
```

Return Value
Type: USBDriver

Example

```
using CipherLab.RFID.Device1800.Config;

USBDriver driver = config.GetUSBDriver();
```

See Also
2.32 USBDriver
Configurator.SetUSBDriver Method

Back to List of Configurator Methods - III

Configurator.IsBluetoothPowerSavingEnabled Method

Gets *Bluetooth*® power saving status.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public bool IsBluetoothPowerSavingEnabled()
```

Return Value
Type: bool

Example

```
using CipherLab.RFID.Device1800.Config;

bool Interface_BluetoothPowerSaving =
config.IsBluetoothPowerSavingEnabled();
```

See Also
Configurator.EnableBluetoothPowerSaving Method

Back to List of Configurator Methods - III

Configurator.IsBluetoothSecureSimplePairingEnabled Method

Gets the value indicating whether *Bluetooth*® Secure Simple Pairing is enabled.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public bool IsBluetoothSecureSimplePairingEnabled()
```

Return Value
Type: bool

Example

```
using CipherLab.RFID.Device1800.Config;

bool enabled = config.IsBluetoothSecureSimplePairingEnabled();
```

See Also
Configurator.EnableBluetoothSecureSimplePairing Method

Back to List of Configurator Methods - III

Configurator.IsBT3610SetToVCOM Method

Gets a value indicating whether the 3610 connection interface is set to VCOM.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public bool IsBT3610SetToVCOM()
```

Return Value

Type: bool
"True" if 3610 is set to VCOM interface. "False" if 3610 is not set to VCOM interface.

Remarks

This method is only available in the Windows version of the SDK.

Example

```
using CipherLab.RFID.Device1800.Config;

bool state = config.IsBT3610SetToVCOM();
```

See Also

Configurator.SetConnectionInterface3610Hid Method
Configurator.SetConnectionInterface3610Vcom Method

Back to List of Configurator Methods - III

Configurator.KeepDeviceAlive Method

Starts a background thread to prevent the 1800 device from entering power-saving mode.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void KeepDeviceAlive()
```

Remarks

The background thread is automatically stopped when the 1800 device is disconnected physically or by calling the connector's Disconnect() method.

Example

```
using CipherLab.RFID.Device1800.Config;

config.KeepDeviceAlive();
```

Back to List of Configurator Methods - III

Configurator.LoadFactoryDefaultSettings Method

Resets the 1800 device to its factory default settings.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void LoadFactoryDefaultSettings()
```

Example

```
using CipherLab.RFID.Device1800.Config;

config.LoadFactoryDefaultSettings();
```

See Also

Configurator.LoadUserSettings Method

Back to List of Configurator Methods - III

Configurator.LoadUserSettings Method

Resets the 1800 device with user setting.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void LoadUserSetting()
```

Example

```
using CipherLab.RFID.Device1800.Config;

config.LoadUserSetting();
```

See Also

Configurator.LoadFactoryDefaultSettings Method
Configurator.SaveUserSettings Method

Back to List of Configurator Methods - III

Configurator.ReadDeviceMemoryData Method

Reads the device memory data.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public List<MemoryData> ReadDeviceMemoryData(
        bool clear
        out int total
)
```

Parameter	Description
clear	Type: bool Clears the device memory upon return.
total	Type: int Total number of records in the device memory.

Return Value

Type: List<MemoryData>
A list of scanned records in the device memory.

Example

```
using CipherLab.RFID.Device1800.Config;

int total;
List<MemoryData> data = config.ReadDeviceMemoryData(false, out total);
```

See Also

3.5 MemoryData
Configurator.ClearDeviceMemoryData Method
Configurator.GetDeviceOperatingMode Method
Configurator.SetDeviceOperatingMode Method

Back to List of Configurator Methods - III

Configurator.ResetDataCounter Method

Resets the tag counter.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void ResetDataCounter()
```

Example

```
using CipherLab.RFID.Device1800.Config;

config. ResetDataCounter ();
```

See Also

Configurator.GetCounterResetOptions Method
Configurator.GetMultiTagModeCounterLimit Method
Configurator.GetMultiTagModeSettings Method
Configurator.SetCounterResetOptions Method
Configurator.SetMultiTagModeCounterLimit Method
Configurator.SetMultiTagModeSettings Method

Back to List of Configurator Methods - III

Configurator.RFIDDirectCancelInventoryRound Method

Stops the 1800 device from tag scanning during an inventory round.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void RFIDDirectCancelInventoryRound()
```

Example

```
using CipherLab.RFID.Device1800.Config;

RFIDDirectCancelInventoryRound();
```

See Also

Configurator.RFIDDirectStartInventoryRound Method

Back to List of Configurator Methods - IV

Configurator.RFIDDirectKillTag Method

Automatically triggers the 1800 device to kill a RFID tag with a given EPC. Trigger-press is not required.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public DeviceResponse RFIDDirectKillTag(
        byte[] password,
        byte[] epc
)
```

Parameter	Description
password	Type: byte[] Tag kill password
epc	Type: byte[] EPC used to pick out the tag to kill

Return value

Type: DeviceResponse

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] password = new byte[] { 0x31, 0x32, 0x33, 0x34 };
byte[] epc = new byte[] { 0x35, 0x15, 0xFD, 0x85, 0x60, 0x08, 0x23, 0x50,
0x00, 0x00, 0x00, 0x02 };

if (RFIDDirectKillTag(password, epc) ==
DeviceResponse.OperationSuccess)
{
    // tag killed
}
else
{
    // handle error
}
```

See Also

2.9 DeviceResponse

Back to List of Configurator Methods - IV

Configurator.RFIDDirectLockTag Method

Automatically triggers the 1800 device to lock a RFID tag memory bank with a given EPC. Trigger-press is not required.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public DeviceResponse RFIDDirectLockTag(
        byte[] password,
        byte[] epc,
        LockTarget targetBank
)
```

Parameter	Description
password	Type: byte[] Tag access password
epc	Type: byte[] EPC used to pick out the tag to lock
targetBank	Type: LockTarget Tag memory bank which will be locked

Return value

Type: DeviceResponse

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] password = new byte[] { 0x31, 0x32, 0x33, 0x34 };
byte[] epc = new byte[] { 0x35, 0x15, 0xFD, 0x85, 0x60, 0x08, 0x23, 0x50,
0x00, 0x00, 0x00, 0x02 };
byte[] data;

if (RFIDDirectLockTag(password, epc, LockTarget.EPC) ==
DeviceResponse.OperationSuccess)
{
    // epc bank locked
}
else
{
    // handle error
}
```

See Also

2.9 DeviceResponse
2.21 LockTarget
Configurator.RFIDDirectPermanentLockTag Method
Configurator.RFIDDirectUnlockTag Method

Back to List of Configurator Methods - IV

Configurator.RFIDDirectPermanentLockTag Method

Automatically triggers the 1800 device to permanently lock a RFID tag memory bank with a given EPC. Trigger-press is not required.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public DeviceResponse RFIDDirectPermanentLockTag(
        byte[] password,
        byte[] epc,
        LockTarget targetBank
)
```

Parameter	Description
password	Type: byte[] Tag access password
epc	Type: byte[] EPC used to pick out the tag to permanently lock
targetBank	Type: LockTarget Tag memory bank which will be permanently locked

Return value

Type: DeviceResponse

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] password = new byte[] { 0x31, 0x32, 0x33, 0x34 };
byte[] epc = new byte[] { 0x35, 0x15, 0xFD, 0x85, 0x60, 0x08, 0x23, 0x50,
0x00, 0x00, 0x00, 0x02 };
byte[] data;

if (RFIDDirectPermanentLockTag(password, epc, LockTarget.EPC) ==
DeviceResponse.OperationSuccess)
{
    // epc bank permanent locked
}
else
{
    // handle error
}
```

See Also

2.9 DeviceResponse
2.21 LockTarget
Configurator.RFIDDirectLockTag Method

Back to List of Configurator Methods - IV

Configurator.RFIDDirectReadTagByEPC Method

Automatically triggers a RFID tag read on the 1800 device with a given EPC. Trigger-press is not required.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void RFIDDirectReadTagByEPC(
        byte[] password,
        byte[] epc,
        RFIDMemoryBank bank,
        int start,
        int length,
        int retry,
        ReadCallBack callback
)
```

Parameter	Description
password	Type: byte[] Tag access password
epc	Type: byte[] EPC used to pick out the tag to read
bank	Type: RFIDMemoryBank Tag memory bank to read from
start	Type: int Start byte position in bank memory
length	Type: int Number of bytes to read
retry	Type: int Number of retries if initial read fails
callback	Type: ReadCallBack Delegate for receiving data read from the given RFID tag memory bank

Return value

Type: void

Example

```
using CipherLab.RFID.Device1800.Config;

private void ReadDataCallback(byte[] data, DeviceResponse response)
{
    // do something with data
}

byte[] password = new byte[] { 0x31, 0x32, 0x33, 0x34 };
byte[] epc = new byte[] { 0x35, 0x15, 0xFD, 0x85, 0x60, 0x08, 0x23, 0x50,
0x00, 0x00, 0x00, 0x02 };
byte[] data;

// read the first byte of TID bank
```

```
// retry 3 times if initial read failed
RFIDDirectReadTagByEPC(password, epc, RFIDMemoryBank.TID, 0, 1, 3,
ReadDataCallback);
```

See Also

2.9 DeviceResponse
Configurator.RFIDDirectReadTagByTID Method
Configurator.RFIDDirectWriteTagByEPC Method

Back to List of Configurator Methods - IV

Configurator.RFIDDirectReadTagByTID Method

Automatically triggers a RFID tag read on the 1800 device with a given TID. Trigger-press is not required.

Note: This method is currently not supported by 1862 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void RFIDDirectReadTagByTID(
        byte[] password,
        byte[] tid,
        RFIDMemoryBank bank,
        int start,
        int length,
        int retry,
        ReadCallBack callback
)
```

Parameter	Description
password	Type: byte[] Tag access password
tid	Type: byte[] TID used to pick out the tag to read
bank	Type: RFIDMemoryBank Tag memory bank to read from
start	Type: int Start byte position in bank memory
length	Type: int Number of bytes to read
retry	Type: int Number of retries if initial read fails
callback	Type: ReadCallBack Delegate for receiving data read from the given RFID tag memory bank

Return value
Type: void

Example

```
using CipherLab.RFID.Device1800.Config;

private void ReadDataCallback(byte[] data, DeviceResponse response)
{
    // do something with data
}

byte[] password = new byte[] { 0x31, 0x32, 0x33, 0x34 };
byte[] tid = new byte[] { 0xE2, 0x00, 0x10, 0x93 };
byte[] data;
```

```
// read the first byte of EPC bank
// retry 3 times if initial read failed
RFIDDirectReadTagByTID(password, tid, RFIDMemoryBank.EPC, 0, 1, 3,
ReadDataCallback);
```

See Also

2.9 DeviceResponse
Configurator.RFIDDirectReadTagByEPC Method
Configurator.RFIDDirectWriteTagByTID Method

Back to List of Configurator Methods - IV

Configurator.RFIDDirectStartInventoryRound Method

Automatically triggers a RFID inventory round on the 1800 device. Trigger-press is not required.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void RFIDDirectStartInventoryRound(
        InventoryType type,
        int count,
        InventoryCallBack callback
)
```

Parameter	Description
type	Type: InventoryType Format of the collected data in the inventory round Note: InventoryType.EPC_andTID is currently not supported by 1862 device.
count	Type: int Number of reads in one inventory round
callback	Type: InventoryCallBack Delegate for receiving data read from the given RFID tag memory bank

Return value

Type: void

Example

```
using CipherLab.RFID.Device1800.Config;

private void InventoryData(InventoryData data, DeviceResponse response)
{
    // do something with data
}

RFIDDirectStartInventoryRound(InventoryType.EPC, 100, InventoryData);
```

See Also

2.9 DeviceResponse
2.17 InventoryType
4.6 EPCData Class
Configurator.RFIDDirectCancelInventoryRound Method

Back to List of Configurator Methods - IV

Configurator.RFIDDirectUnlockTag Method

Automatically triggers the 1800 device to unlock a RFID tag memory bank with a given EPC. Trigger-press is not required.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public DeviceResponse RFIDDirectUnlockTag(
        byte[] password,
        byte[] epc,
        LockTarget targetBank
)
```

Parameter	Description
password	Type: byte[] Tag access password
epc	Type: byte[] EPC used to pick out the tag to unlock
targetBank	Type: LockTarget Tag memory bank which will be unlocked

Return value

Type: DeviceResponse

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] password = new byte[] { 0x31, 0x32, 0x33, 0x34 };
byte[] epc = new byte[] { 0x35, 0x15, 0xFD, 0x85, 0x60, 0x08, 0x23, 0x50,
0x00, 0x00, 0x00, 0x02 };
byte[] data;

if (RFIDDirectUnlockTag(password, epc, LockTarget.EPC) ==
DeviceResponse.OperationSuccess)
{
    // epc bank unlocked
}
else
{
    // handle error
}
```

See Also

2.9 DeviceResponse
2.21 LockTarget
Configurator.RFIDDirectLockTag Method

Back to List of Configurator Methods - IV

Configurator.RFIDDirectWriteTagByEPC Method

Automatically triggers a RFID tag write on the 1800 device with a given EPC. Trigger-press is not required.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public DeviceResponse RFIDDirectWriteTagByEPC(
        byte[] password,
        byte[] epc,
        RFIDMemoryBank bank,
        int start,
        int retry,
        byte[] data
)
```

Parameter	Description
password	Type: byte[] Tag access password
epc	Type: byte[] EPC used to pick out the tag to write
bank	Type: RFIDMemoryBank Tag memory bank to write to
start	Type: int Start byte position in bank memory
retry	Type: int Number of retries if initial write fails
data	Type: byte[] Data to write to the given RFID tag memory bank

Return value

Type: DeviceResponse

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] password = new byte[] { 0x31, 0x32, 0x33, 0x34 };
byte[] epc = new byte[] { 0x35, 0x15, 0xFD, 0x85, 0x60, 0x08, 0x23, 0x50,
0x00, 0x00, 0x00, 0x02 };
byte[] data = new byte[] { 0x35, 0x15, 0xFD, 0x85, 0x60, 0x08, 0x23, 0x50,
0x00, 0x00, 0x00, 0x01 };

if (RFIDDirectWriteTagByEPC(password, epc, RFIDMemoryBank.EPC, 0,  3,
data) == DeviceResponse.OperationSuccess)
{
    // success
}
else
{
    // handle error
}
```

See Also

2.9 DeviceResponse
Configurator.RFIDDirectReadTagByEPC Method
Configurator.RFIDDirectWriteTagByTID Method

Back to List of Configurator Methods - IV

Configurator.RFIDDirectWriteTagByTID Method

Automatically triggers a RFID tag write on the 1800 device with a given TID. Trigger-press is not required.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public DeviceResponse RFIDDirectWriteTagByTID(
        byte[] password,
        byte[] tid,
        RFIDMemoryBank bank,
        int start,
        int retry,
        byte[] data
)
```

Parameter	Description
password	Type: byte[] Tag access password
tid	Type: byte[] TID used to pick out the tag to write
bank	Type: RFIDMemoryBank Tag memory bank to write to
start	Type: int Start byte position in bank memory
retry	Type: int Number of retries if initial write fails
data	Type: byte[] Data to write to the given RFID tag memory bank

Return value

Type: DeviceResponse

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] password = new byte[] { 0x31, 0x32, 0x33, 0x34 };
byte[] tid = new byte[] { 0xE2, 0x00, 0x10, 0x93 };
byte[] data = new byte[] { 0x35, 0x15, 0xFD, 0x85, 0x60, 0x08, 0x23, 0x50,
0x00, 0x00, 0x00, 0x01};


if (RFIDDirectWriteTagByTID(password, tid, RFIDMemoryBank.EPC, 0, 3,
data) == DeviceResponse.OperationSuccess)
{
    // success
}
else
{
    // handle error
}
```

See Also

2.9 DeviceResponse
Configurator.RFIDDirectReadTagByTID Method
Configurator.RFIDDirectWriteTagByEPC Method

Back to List of Configurator Methods - IV

Configurator.RFIDReadTagMassive Method

Prepares the 1800 device for trigger-pressed read.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void RFIDReadTagMassive(
        byte[] password,
        RFIDMemoryBank bank,
        int start,
        int length
)
```

Parameter	Description
password	Type: byte[] Tag access password
bank	Type: RFIDMemoryBank Tag memory bank to read from
start	Type: int Start byte position in bank memory
length	Type: int Number of bytes to read

Example

```
using CipherLab.RFID.Device1800.Config;

config.RFIDReadTagMassive(NULL, RFIDMemoryBank.User, 0,0);
```

See Also

2.26 RFIDMemoryBank
2.27 RFIDMode
Configurator.GetRFIDMode Method
Configurator.SetRFIDMode Method
Configurator.RFIDWriteTagMassive Method

Back to List of Configurator Methods - IV

Configurator.RFIDWriteTagEPCBank Method

Prepares the 1800 device for trigger-pressed EPC bank write.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void RFIDWriteTagEPCBank(
        byte[] password,
        byte[] pc,
        byte[] epc
)
```

Parameter	Description
password	Type: byte[] Tag access password.
pc	Type: byte[] Protocol Control. Set to null to write EPC only.
epc	Type: byte[] EPC to write to the EPC memory bank. Set to null to write PC only.

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] password = new byte[] { 0x31, 0x32, 0x33, 0x34 };
byte[] epc = new byte[] { 0x35, 0x15, 0xFD, 0x85, 0x60, 0x08, 0x23, 0x50,
0x00, 0x00, 0x00, 0x02 };

RFIDWriteTagEPCBank(password, null, epc);
```

See Also

Configurator.RFIDWriteTagMassive Method
Configurator.RFIDWriteTagPassword Method

Back to List of Configurator Methods - IV

Configurator.RFIDWriteTagMassive Method

Prepares the 1800 device for trigger-pressed write.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void RFIDWriteTagMassive(
        byte[] password,
        RFIDMemoryBank bank,
        byte[] data,
        int start,
        int length
)
```

Parameter	Description
password	Type: byte[] Tag access password
bank	Type: RFIDMemoryBank Tag memory bank to write to
data	Type: byte[] Data to write to the given RFID tag memory bank
start	Type: int Start byte position in bank memory
length	Type: int Number of bytes to write

Example

```
using CipherLab.RFID.Device1800.Config;

config.RFIDWriteTagMassive(NULL, RFIDMemoryBank.User, "Test", 0, 4);
```

See Also

2.26 RFIDMemoryBank
2.27 RFIDMode
Configurator.GetRFIDMode Method
Configurator.SetRFIDMode Method
Configurator.RFIDReadTagMassive Method
Configurator.RFIDWriteTagEPCBank Method
Configurator.RFIDWriteTagPassword Method

Back to List of Configurator Methods - IV

Configurator.RFIDWriteTagPassword Method

Prepares the 1800 device for trigger-pressed reserved bank write.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void RFIDWriteTagPassword(
        byte[] accessPassword,
        byte[] newAccessPassword,
        byte[] newKillPassword
)
```

Parameter	Description
accessPassword	Type: byte[] Tag access password.
newAccessPassword	Type: byte[] New access password. Set to null if you only want to write the kill password.
newKillPassword	Type: byte[] New kill password. Set to null if you only want to write the access password.

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] password = new byte[] { 0x31, 0x32, 0x33, 0x34 };

// clears both access and kill passwords
RFIDWriteTagPassword(password, Configurator.EmptyPassword,
Configurator.EmptyPassword);
```

See Also

Configurator.RFIDWriteTagMassive Method
Configurator.RFIDWriteTagEPCBank Method

Back to List of Configurator Methods - IV

Configurator.SaveUserSettings Method

Saves the current 1800 device setting as the user setting.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SaveUserSettings()
```

Example

```
using CipherLab.RFID.Device1800.Config;

config.SaveUserSettings();
```

See Also

LoadFactoryDefaultSettings Method
LoadUserSettings Method

Back to List of Configurator Methods - IV

Configurator.SelectMemoryBank Method

Selects a memory bank to read or write.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SelectMemoryBank(
        RFIDMemoryBank bank
)
```

Return Value
Type: RFIDMemoryBank

Example

```
using CipherLab.RFID.Device1800.Config;

config. SeletMemoryBank (RFIDMemoryBank.User);
```

See Also
2.26 RFIDMemoryBank
Configurator.GetSelectedMemoryBank Method

Back to List of Configurator Methods - IV

Configurator.SetBluetoothInterfaceType Method

Changes the current *Bluetooth*® Interface.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetBluetoothInterfaceType(
        BluetoothInterface BTInterface
)
```

Parameter	Description
devNotif	Type: BluetoothInterface See **BluetoothInterface** enumeration for a list of Bluetooth® connection interface supported by the 1800 device.

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetBluetoothInterfaceType(BluetoothInterface.SPP_Slave);
```

See Also

2.4 BluetoothInterface
Configurator.GetBluetoothInterfaceType Method

Back to List of Configurator Methods - IV

Configurator.SetConnectionInterface3610Hid Method

Configures the 3610 HID connection settings of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetConnectionInterface3610Hid(
        HID hid,
        string serialNumber
)
```

Parameter	Description
hid	Type: HID 3610 HID settings
serialNumber	Type: string The serial number of CipherLab 3610 dongle has 9 characters.

Remarks

This method is only available in the Windows version of the SDK.
It will cause the BT module of the 1800 device to re-initialize, which takes approximately 3.6 seconds.

Example

```
using CipherLab.RFID.Device1800.Config;

HID hid = new HID();
config. SetConnectionInterface3610Hid (hid, "BS9001346");
```

See Also

3.4 HID Configurator.GetConnectionInterface3610Hid Method

Back to List of Configurator Methods - IV

Configurator.SetConnectionInterface3610Vcom Method

Sets the serial number of the CipherLab 3610 dongle which you'd like to connect your 1800 device to. (3610 should be connected to PC USB port)

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetConnectionInterface3610Vcom(
        string serialNumber
)
```

Parameter	Description
serialNumber	Type: string The serial number of CipherLab 3610 dongle has 9 characters.

Remarks

This method is only available in the Windows version of the SDK.
It will cause the BT module of the 1800 device to re-initialize, which takes approximately 3.6 seconds.

Example

```
using CipherLab.RFID.Device1800.Config;

string serialNumber = "BS9001346";
bool r = config. SetConnectionInterface3610Vcom (serialNumber);
```

See Also

Configurator.GetConnectionInterface3610Vcom Method

Back to List of Configurator Methods - IV

Configurator.SetConnectionInterfaceHid Method

Configures the *Bluetooth*® HID connection settings of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetConnectionInterfaceHid(
        bool visible,
        bool secured,
        string pin,
        HID hid
)
```

Parameter	Description
visible	Type: bool A value indicating whether the 1800 device can be discovered by other Bluetooth® devices
secured	Type: bool A value indicating whether Bluetooth® authentication is enabled for the 1800 device.
pin	Type: string Bluetooth® PIN for pairing: 1~16 digits **Default PIN=0000**
hid	Type: HID Bluetooth® HID settings

Example

```
using CipherLab.RFID.Device1800.Config;

HID hid = new HID();
hid.KbdType =  HIDKeyboardType.PCAT_US;
hid.InterCharDelay = 100;
hid.LockState = CapsLockState.On;
hid.LockType = CapsLockType.Normal;
hid.IgnoreCaseOnAlphabetsTransmission = true;
hid.DigitsTransmissionKeypadType = KeypadType.AlphaNumericKeypad;
hid.DigitsPos = DigitPosition.Normal;
hid.KbdLayout = KeyboardLayout.Normal;
hid.HidCharTransmitMode = HIDCharTransmitMode.ByCharacter;
config.SetConnectionInterfaceHid(true, true, "0000", hid);
```

Remark

This method changes the connection interface of the 1800 device to Bluetooth® HID and disconnects the 1800 device from the connected terminal device. It will cause the BT module of the 1800 device to re-initialize, which takes approximately 3.6 seconds.

See Also

2.10 DigitPosition,
2.15 HIDCharTransmitMode
2.16 HIDKeyboardType
2.19 KeyboardLayout

2.20 KeypadType Configurator.GetConnectionInterfaceHid Method

Back to List of Configurator Methods - V

Configurator.SetConnectionInterfaceSppMaster Method

Configures the *Bluetooth*® SPP Master connection settings of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetConnectionInterfaceSppMaster(
        string btAddr,
        bool visible,
        bool secured,
        string pin
)
```

Parameter	Description
btAddr	Type: string Hexadecimal representation of the Bluetooth® address of device or computer to connect to without separators.
visible	Type: bool A value indicating whether the 1800 device can be discovered by other Bluetooth® devices
secured	Type: bool A value indicating whether Bluetooth® authentication is enabled for the 1800 device.
pin	Type: string Bluetooth® PIN for pairing: 1~16 digits **Default PIN=0000**

Remarks

This method is only available in the Windows version of the SDK.
It will cause the BT module of the 1800 device to re-initialize, which takes approximately 3.6 seconds.
Bluetooth address will not be overwritten until a successful connection has been established.

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetConnectionInterfaceSppMaster("0123456789ab", true, true,
"1234");
```

See Also

Configurator.DisconnectBluetooth Method
Configurator.GetConnectionInterfaceSppMaster Method

Back to List of Configurator Methods - V

Configurator.SetConnectionInterfaceSppSlave Method

Configures the *Bluetooth*® SPP Slave connection settings of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetConnectionInterfaceSppSlave(
        bool visible,
        bool secured,
        string pin
)
```

Parameter	Description
visible	Type: bool A value indicating whether the 1800 device can be discovered by other Bluetooth® devices
secured	Type: bool A value indicating whether Bluetooth® authentication is enabled for the 1800 device.
pin	Type: string Bluetooth® PIN for pairing: 1~16 digits **Default PIN=0000**

Remarks

This method will cause the BT module of the 1800 device to re-initialize, which takes approximately 3.6 seconds.

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetConnectionInterfaceSppSlave(true, true, "1234");
```

See Also

Configurator.GetConnectionInterfaceSppSlave Method

Back to List of Configurator Methods - V

Configurator.SetCounterFormat Method (byte[], byte[], char)

Sets counter format.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetCounterFormat(
        byte[] prefix,
        byte[] suffix,
        char leftPaddingChar
)
```

Parameter	Description
prefix	Type: byte[] Counter prefix in bytes. Max. 8 bytes allowed. Null to disable counter prefix.
suffix	Type: byte[] Counter suffix in bytes. Max. 8 bytes allowed. Null to disable counter suffix.
leftPaddingChar	Type: char A character to pad before counter.

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] prefix = { 0x20, 0x42};
byte[] suffix = {0x0D};
char leftPaddingChar = '#';
config.SetCounterFormat(prefix,suffix,leftPaddingChar);
```

See Also

2.7 DataSequenceSection
Configurator.GetCounterFormat Method (byte[], byte[], char)
Configurator.GetCounterFormat Method (string, string, char)
Configurator.SetCounterFormat Method (string, string, char)

Back to List of Configurator Methods - V

Configurator.SetCounterFormat Method (string, string, char)

Sets the counter format.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetCounterFormat(
        string prefix,
        string suffix,
        char leftPaddingChar
)
```

Parameter	Description
prefix	Type: string Counter prefix in string format. Max. 8 characters allowed.
suffix	Type: string Counter suffix in string format. Max. 8 characters allowed.
leftPaddingChar	Type: char A character to pad before counter.

Example

```
using CipherLab.RFID.Device1800.Config;

string prefix = "@%";
string suffix = "**";
char leftPaddingChar = '#';
config.SetCounterFormat(prefix,suffix,leftPaddingChar);
```

See Also

2.7 DataSequenceSection
Configurator.GetCounterFormat Method (byte[], byte[], char)
Configurator.GetCounterFormat Method (string, string, char)
Configurator.SetCounterFormat Method (byte[], byte[], char)

Back to List of Configurator Methods - V

Configurator.SetCounterResetOptions Method

Sets the times to reset the counter on the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetCounterResetOptions(
        bool byCommand,
        bool rfidPowerOn,
        bool newBTConnection
)
```

Parameter	Description
byCommand	Type: bool A value indicating whether the counter is reset when a reset command is received.
rfidPowerOn	Type: bool A value indicating whether the counter is reset when the RFID module powers on.
newBTConnection	Type: bool A value indicating whether the counter is reset when a new Bluetooth® connection is established

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetCounterResetOptions(True ,True, True);
```

See Also

Configurator.GetCounterResetOptions Method
Configurator.GetMultiTagModeCounterLimit Method
Configurator.GetMultiTagModeSettings Method
Configurator.ResetDataCounter Method
Configurator.SetMultiTagModeCounterLimit Method
Configurator.SetMultiTagModeSettings Method

Back to List of Configurator Methods - V

Configurator.SetDataPortal Method

Sets the 1800 device's data outlet.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetDataPortal(
        DataPortal portal
)
```

Parameter	Description
portal	Type: DataPortal Scanned data outlet.

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetDataPortal(DataPortal.Bluetooth);
```

See Also

2.6 DataPortal
Configurator.GetDataPortal Method

Back to List of Configurator Methods - V

Configurator.SetDeviceNotificationState Method

Changes the notification settings of the 1800 device, including system and good read beep, low battery alert, LED indication and vibration.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetDeviceNotificationState(
        DeviceNotification devNotif
)
```

Parameter	Description
devNotif	Type: DeviceNotification

Example

```
using CipherLab.RFID.Device1800.Config;

DeviceNotification nor = new DeviceNotification();
nor.LowBatteryAlertEnabled = true;
nor.LEDEnabled = true;
nor.LEDDuration = 4;
nor.VibratorEnabled = true;
nor.VibratorDuration = 10;
nor.GoodReadBeeperEnabled = true
nor.BeeperVolume = BeepVolume.High;
nor.BeeperFrequency = BeepFrequency.Freq_1kHz;
nor.BeeperDuration = BeepDuration.Long;
config.SetDeviceNotificationState(nor);
```

See Also

3.2 DeviceNotification
Configurator.GetDeviceNotificationState Method

Back to List of Configurator Methods - V

Configurator.SetDeviceOperatingMode Method

Changes the 1800 device operating mode.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetDeviceOperatingMode(
        OperatingMode mode
)
```

Parameter	Description
mode	Type: OperatingMode See **OperatingMode** enumeration for the modes

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetDeviceOperatingMode(OperatingMode.Online);
```

See Also

2.23 OperatingMode
Configurator.GetDeviceOperatingMode Method

Back to List of Configurator Methods - V

Configurator.SetDevicePowerSavingState Method

Sets the power saving timeouts of the 1800 device. By default, the idle power saving timeout is set to 2 minutes and the idle shutdown timeout is set to 10 minutes.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetDevicePowerSavingState(
        int standby,
        int shutdown
)
```

Parameter	Description
standby	Type: int Idle minutes to enter power saving mode. Maximum value is 254 minutes. If set to 0, the 1800 device never enters power saving mode.
shutdown	Type: int Idle minutes to shut down the device. Maximum value is 254 minutes. If set to 0, the 1800 idle shutdown is disabled.

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetDevicePowerSavingState(100,50);
```

See Also

Configurator.GetDevicePowerSavingState Method
Configurator.ShutdownDevice Method

Back to List of Configurator Methods - V

Configurator.SetDeviceTime Method

Changes the 1800 device clock. Year format can be specifed as four-digit year (e.g. 2012) or two-digit year(e.g. 12).

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetDeviceTime(
        DateTime dateTime
)
```

Parameter	Description
dateTime	Type: DateTime

Example

```
using CipherLab.RFID.Device1800.Config;

DateTime currect = DateTime.Now;
config.SetDeviceTime(current);
```

See Also

Configurator.GetDeviceTime Method

Back to List of Configurator Methods - V

Configurator.SetEPCFilter Method

Sets how the 1800 device filters RFID tags.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetEPCFilter(
        EPCFilter filter
)
```

Parameter	Description
filter	Type: EPCFilter See **EPCFilter** structure.

Example

```
using CipherLab.RFID.Device1800.Config;

Example 1:
EPCFilter filter = new EPCFilter();
filter.Enabled = true;
filter.Encoding = EPCEncodingScheme.SGTIN96;
filter.FilterType = EPCFilterType.IncludePattern;
filter.FilterMethod = FilterMethod.FilterValue;
filter.FilterValue = 6;
config.SetEPCFilter(filter);

Example 2:
EPCFilter filter = new EPCFilter();
filter.Enabled = true;
filter.Encoding = EPCEncodingScheme.SGTIN96;
filter.FilterType = EPCFilterType.IncludePattern;
filter.FilterMethod = FilterMethod.Partition1;
filter.Part1Value = 123456;
config.SetEPCFilter(filter);

example 03:
EPCFilter filter = new EPCFilter();
filter.Enabled = true;
filter.Encoding = EPCEncodingScheme.SGTIN96;
filter.FilterType = EPCFilterType.IncludePattern;
filter.FilterMethod = FilterMethod.Partition2;
filter.Part2FromValue = 123;
filter.Part2ToValue = 456;
filter.Part2MaxLength = 6;
config.SetEPCFilter(filter);
```

See Also

2.12 EPCFilterType
2.13 FilterMethod
3.3 EPCFilter

GetEPCFilter Method

Back to List of Configurator Methods - V

Configurator.SetFirmwareUpdateInterface Method

Sets firmware update interface of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetFirmwareUpdateInterface(
        FirmwareUpdateInterface interfaceType
)
```

Parameter	Description
interfaceType	Type: FirmwareUpdateInterface

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetFirmwareUpdateInterface(FirmwareUpdateInterface.USB);
```

See Also

2.14 FirmwareUpdateInterface

Back to List of Configurator Methods - V

Configurator.SetKeyActionString Method

Sets the 1800 device hardware keys actions and the corresponding strings that will be sent when the key action is triggered.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetKeyActionString(
        KeyActionString[] keyActStr
)
```

Parameter	Description
keyActStr	Type: KeyActionString[] See **KeyActionString** structure.

Example

```
using CipherLab.RFID.Device1800.Config;

KeyActionString[] action = new KeyActionString[6];
action[0].action = KeyAction.F1Down;
action[0].text = "#@TRIGON\r";
action[1].action = KeyAction.F2Down;
action[1].text = "#@TRIGOFF\r";
config.SetKeyActionString(action);
```

See Also

2.18 KeyAction
3.7 KeyActionString
GetKeyActionString Method

Back to List of Configurator Methods - V

Configurator.SetMemoryModeDelay Method

Sets the delay time of Memory mode data transmission.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetMemoryModeDelay(
        TransmissionDelay bufferDelay
)
```

Return Value

Type: TransmissionDelay
See **TransmissionDelay** enumeration for a list of delay values.

Example

```
using CipherLab.RFID.Device1800.Config;

TransmissionDelay DataTransmissionDelay =
TransmissionDelay.Delay_1sec
config.SetMemoryModeDelay(System_DataTransmissionDelay);
```

See Also

Configurator.GetMemoryModeDelay Method
Configurator.SetDeviceOperatingMode Method

Back to List of Configurator Methods - V

Configurator.SetMultiTagListType Method

Sets the data type in which the multi-tag list of the 1800 device should store in order to avoid duplicate tag scans.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetMultiTagListType (
        MultiTagListType type
)
```

Parameter	Description
type	Type: MultiTagListType See **MultiTagListType** enumeration.

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetMultiTagListType(MultiTagListType.EPC);
```

See Also

2.22 MultiTagListType
GetMultiTagListType Method

Back to List of Configurator Methods - V

Configurator.SetMultiTagModeCounterLimit Method

Changes the counter limit of current multi-tag settings.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetMultiTagModeCounterLimit (
        int limit
)
```

Parameter	Description
limit	Type: int Limit of the multi-tag mode counter.

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetMultiTagModeCounterLimit(128);
```

Remarks

Calls to this method will clear the unique tag list of the 1800 device.

See Also

Configurator.GetCounterResetOptions Method
Configurator.GetMultiTagModeCounterLimit Method
Configurator.GetMultiTagModeSettings Method
Configurator.ResetDataCounter Method
Configurator.SetCounterResetOptions Method
Configurator.SetMultiTagModeSettings Method

Back to List of Configurator Methods - V

Configurator.SetMultiTagModeSettings Method

Changes the multi-tag settings of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetMultiTagModeSettings(
        ScanDelay scanDelay,
        int counterLimit,
        bool warnLimitReached
)
```

Parameter	Description
scanDelay	Type: ScanDelay See **ScanDelay** enumeration for a list of delay time intervals.
counterLimit	Type: int Limit of multi-tag mode counter for avoiding duplicate tag scans. The legal counter range is from 0 to 128. When the counter is set to 0, duplicate tag scan is allowed.
warnLimitReached	Type: bool Beep if the multi-tag list is full

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetMultiTagModeSettings(ScanDelay.Delay_100ms, 0, false);
```

Remarks

Calls to this method will clear the unique tag list of the 1800 device.

See Also

2.28 ScanDelay
Configurator.GetCounterResetOptions Method
Configurator.GetMultiTagModeCounterLimit Method
Configurator.GetMultiTagModeSettings Method
Configurator.ResetDataCounter Method
Configurator.SetCounterResetOptions Method
Configurator.SetMultiTagModeCounterLimit Method

Back to List of Configurator Methods - VI

Configurator.SetOnlineModeDelay Method

Changes the delay time of online mode data transmission.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetOnlineModeDelay(
        TransmissionDelay bufferDelay
)
```

Parameter	Description
bufferDelay	Type: TransmissionDelay See **TransmissionDelay** enumeration for a list of delay values.

Example

```
using CipherLab.RFID.Device1800.Config;

TransmissionDelay System_TransmissionDelay =
TransmissionDelay.Delay_1sec
config.SetOnlineModeDelay(System_TransmissionDelay);
```

See Also

2.31 TransmissionDelay
Configurator.GetOnlineModeDelay Method

Back to List of Configurator Methods - VI

Configurator.SetOutputDataFormat Method

Sets the data output data format of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetOutputDataFormat(
        OutputDataFormat dataFormat
)
```

Parameter	Description
dataFormat	Type: OutputDataFormat See **OutputDataFormat** enumeration for a list of supported formats.

Example

```
using CipherLab.RFID.Device1800.Config;

OutputDataFormat OUTPUT_Format = OutputDataFormat.PacketData
config.SetOutputDataFormat(OUTPUT_Format);
```

See Also

2.24 OutputDataFormat
Configurator.GetOutputDataFormat Method

Back to List of Configurator Methods - VI

Configurator.SetOutputDataSequence Method

Sets the sequence of output data.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetOutputDataSequence(
        DataSequenceSection section1,
        DataSequenceSection section2,
        DataSequenceSection section3
)
```

Parameter	Description
section1	Type: DataSequenceSection See **DataSequenceSection** enumeration.
section2	Type: DataSequenceSection See **DataSequenceSection** enumeration.
section3	Type: DataSequenceSection See **DataSequenceSection** enumeration.

Example

```
using CipherLab.RFID.Device1800.Config;

DataSequenceSection section1 = DataSequenceSection.Count;
DataSequenceSection section2 = DataSequenceSection.Timestamp;
DataSequenceSection section3 = DataSequenceSection.Data;
config.SetOutputDataSequence(section1, section2, section3);
```

See Also

2.7 DataSequenceSection
Configurator.GetOutputDataSequence Method

Back to List of Configurator Methods - VI

Configurator.SetQValue Method

Sets the Q value.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetQValue(
        int Q
)
```

Parameter	Description
Q	Type: int The Q value (must be in the range of 0 to 15).

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetQValue(0);
```

Remark

This method changes the Q value, which defines the number of slots in which the tag places the answer randomly. Higher Q values should be used when there are several tags to be read.
For every value of "Q", the reader searches 2^q slots in every inventory round, and for as many cycles until, all the tags are exhausted. A controlled number of tags respond in every round so as to avoid many collisions and also to minimize the number of rounds required to inventories the whole tag population.

See Also

Configurator.GetQValue Method

Back to List of Configurator Methods - VI

Configurator.SetRecognizedEPCEncoding Method

Sets the EPC encoding schemes that are recognized by the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetRecognizedEPCEncoding(
        EPCEncodingScheme encoding
)
```

Parameter	Description
encoding	Type: EPCEncodingScheme

Example

```
using CipherLab.RFID.Device1800.Config;

config. SetRecognizedEPCEncoding(EPCEncodingScheme.GDTI96|
EPCEncodingScheme.GSRN96);
```

See Also

2.11 EPCEncodingScheme
Configurator.GetRecognizedEPCEncoding Method

Back to List of Configurator Methods - VI

Configurator.SetRFIDDataFormat Method (byte[], byte[], RFIDData)

Sets RFID data format.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetRFIDDataFormat(
        byte[] prefix,
        byte[] suffix,
        RFIDData dataFormat
)
```

Parameter	Description
prefix	Type: byte[] Data prefix in bytes. Max. 8 bytes allowed. Null to disable data prefix.
suffix	Type: byte[] Data suffix in bytes. Max. 8 bytes allowed. Null to disable data suffix.
dataFormat	Type: RFIDData See **RFIDData** structure.

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] prefix = { 0x20, 0x42};
byte[] suffix = {0x0D};
RFIDData data = new RFIDData();
data.Field[0] = RFIDDataField.PC;
data.Field[1] = RFIDDataField.EPC;
data.Field[2] = RFIDDataField.CRC;
data.Field[3] = RFIDDataField.MemoryData;
data.Field[4] = RFIDDataField.NotUsed;
data.Separator[0] = (char)0;
data.Separator[1] = (char)0;
data.Separator[2] = (char)0;
data.Separator[3] = (char)0;
config.SetRFIDDataFormat(prefix, suffix, data);
```

See Also

2.7 DataSequenceSection
2.25 RFIDDataField
3.6 RFIDData
Configurator.GetRFIDDataFormat Method (byte[], byte[], RFIDData)
Configurator.GetRFIDDataFormat Method (string, string, RFIDData)
Configurator.SetRFIDDataFormat Method (string, string, RFIDData)

Back to List of Configurator Methods - VI

Configurator.SetRFIDDataFormat Method (string, string, RFIDData)

Sets RFID data format.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetRFIDDataFormat(
        string prefix,
        string suffix,
        RFIDData dataFormat
)
```

Parameter	Description
prefix	Type: string Data prefix in string format. Max. 8 characters allowed.
suffix	Type: string Data suffix in string format. Max. 8 characters allowed.
dataFormat	Type: RFIDData See **RFIDData** structure.

Example

```
using CipherLab.RFID.Device1800.Config;

string prefix = "@%";
string suffix = "**";
RFIDData data = new RFIDData();
data.Field[0] = RFIDDataField.PC;
data.Field[1] = RFIDDataField.EPC;
data.Field[2] = RFIDDataField.CRC;
data.Field[3] = RFIDDataField.MemoryData;
data.Field[4] = RFIDDataField.NotUsed;
data.Separator[0] = (char)0;
data.Separator[1] = (char)0;
data.Separator[2] = (char)0;
data.Separator[3] = (char)0;
config.SetRFIDDataFormat(prefix, suffix, data);
```

See Also

2.7 DataSequenceSection
2.25 RFIDDataField
3.6 RFIDData
Configurator.GetRFIDDataFormat Method (byte[], byte[], RFIDData)
Configurator.GetRFIDDataFormat Method (string, string, RFIDData)
Configurator.SetRFIDDataFormat Method (byte[], byte[], RFIDData)

Back to List of Configurator Methods - VI

Configurator.SetRFIDMode Method

Sets the RFID scan mode of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetRFIDMode(
        RFIDMode mode
)
```

Parameter	Description
mode	Type: RFIDMode See **RFIDMode** enumeration.

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetRFIDMode(RFIDMode.WriteTag);
```

See Also

2.27 RFIDMode
Configurator.GetRFIDMode Method

Back to List of Configurator Methods - VI

Configurator.SetRFIDPowerLevel Method

Sets the 1800 device RFID module signal strength level.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetRFIDPowerLevel(
        int level
)
```

Parameter	Description
level	Type: int Minimum value: 0 Maximum value: 3

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetRFIDPowerLevel(3);
```

See Also

Configurator.GetRFIDPowerLevel Method

Back to List of Configurator Methods - VI

Configurator.SetScanMode Method

Changes the current scan mode.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetScanMode(
        ScanMode mode
)
```

Return Value
Type: ScanMode
See **ScanMode** enumeration for a list of modes.

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetScanMode(ScanMode.SingleTag);
```

See Also
2.29 ScanMode
Configurator.GetScanMode Method

Back to List of Configurator Methods - VI

Configurator.SetSingleTagModeSessionTimeout Method

Sets current scan session timeout in single tag scan mode.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetSingleTagModeTimeout(
        int sessionTimeout
)
```

Parameter	Description
sessionTimeout	Type: int Scan session timeout in seconds, range=0~254 (sec). Default value is 0.

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetSingleTagModeTimeout(254);
```

See Also

Configurator.GetScanMode Method
Configurator.GetSingleTagModeSessionTimeout Method
Configurator.SetScanMode Method

Back to List of Configurator Methods - VI

Configurator.SetTimestampFormat Method (byte[], byte[], Timestamp)

Sets timestamp format.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetTimestampFormat(
        byte[] prefix,
        byte[] suffix,
        Timestamp timestampFormat
)
```

Parameter	Description
prefix	Type: byte[] Timestamp prefix in bytes. Max. 8 bytes allowed. Null to disable timestamp prefix.
suffix	Type: byte[] Timestamp suffix in bytes. Max. 8 bytes allowed. Null to disable timestamp suffix.
timestampFormat	Type: Timestamp See **Timestamp** structure.

Example

```
using CipherLab.RFID.Device1800.Config;

byte[] prefix = { 0x20, 0x42};
byte[] suffix = {0x0D};
Timestamp timestamp = new Timestamp();
timestamp.Field[0] = TimestampField.Year;
timestamp.Field[1] = TimestampField.Month;
timestamp.Field[2] = TimestampField.Day;
timestamp.Field[3] = TimestampField.NotUsed;
timestamp.Field[4] = TimestampField.Hour;
timestamp.Field[5] = TimestampField.Minute;
timestamp.Field[6] = TimestampField.Second;
timestamp.Separator[0] = '-';
timestamp.Separator[1] = '-';
timestamp.Separator[2] = ' ';
timestamp.Separator[3] = ' ';
timestamp.Separator[4] = ':';
timestamp.Separator[5] = ':';
config.SetTimestampFormat(prefix, suffix, timestamp);
```

See Also

2.7 DataSequenceSection
2.30 TimestampField
3.8 Timestamp
Configurator.GetTimestampFormat Method (byte[], byte[], Timestamp)
Configurator.GetTimestampFormat Method (string, string, Timestamp)
Configurator.SetTimestampFormat Method (string, string, Timestamp)

Back to List of Configurator Methods - VI

Configurator.SetTimestampFormat Method (string, string, Timestamp)

Sets the format of timestamp.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetTimestampFormat(
        string prefix,
        string suffix,
        Timestamp timestampFormat
)
```

Parameter	Description
prefix	Type: string Timestamp prefix in string format. Max. 8 characters allowed.
suffix	Type: string Timestamp suffix in in string format. Max. 8 characters allowed.
timestampFormat	Type: Timestamp See **Timestamp** structure.

Example

```
using CipherLab.RFID.Device1800.Config;

string prefix = "";
string suffix = "";
Timestamp timestamp = new Timestamp();
timestamp.Field[0] = TimestampField.Year;
timestamp.Field[1] = TimestampField.Month;
timestamp.Field[2] = TimestampField.Day;
timestamp.Field[3] = TimestampField.NotUsed;
timestamp.Field[4] = TimestampField.Hour;
timestamp.Field[5] = TimestampField.Minute;
timestamp.Field[6] = TimestampField.Second;
timestamp.Separator[0] = '-';
timestamp.Separator[1] = '-';
timestamp.Separator[2] = ' ';
timestamp.Separator[3] = ' ';
timestamp.Separator[4] = ':';
timestamp.Separator[5] = ':';
config.SetTimestampFormat(prefix, suffix, timestamp);
```

Remarks

To set a timestamp field to second, use either TimestampField.Second or TimestampField.SecondWithMillisecond but not both. Using both TimestampField.Second and TimestampField.SecondWithMillisecond in a timestamp would cause an error.

See Also

2.7 DataSequenceSection
2.30 TimestampField
3.8 Timestamp
Configurator.GetTimestampFormat Method (byte[], byte[], Timestamp)
Configurator.GetTimestampFormat Method (string, string, Timestamp)
Configurator.SetTimestampFormat Method (byte[], byte[], Timestamp)

Back to List of Configurator Methods - VI

Configurator.SetUSBDriver Method

Sets the current USB driver in use.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void SetUSBDriver(
        USBDriver driver
)
```

Parameter	Description
driver	Type: USBDriver See **USBDriver** enumeration.

Example

```
using CipherLab.RFID.Device1800.Config;

config.SetUSBDriver(USBDriver.CDC);
```

See Also

2.32 USBDriver
Configurator.GetUSBDriver Method

Back to List of Configurator Methods - VI

Configurator.ShutdownDevice Method

Shuts down the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public void ShutdownDevice()
```

Example

```
using CipherLab.RFID.Device1800.Config;

config.ShutdownDevice();
```

See Also

Configurator.SetDevicePowerSavingState Method

Back to List of Configurator Methods - VI

4.5 BTDevice

The **BTDevice** class contains *Bluetooth*® device information.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public class BTDevice
{
        public string Name
        public byte[] Address
}
```

The BTDevice type exposes the following members.

Fields

	Name	Description
public	Name	Name of the Bluetooth® device
public	Address	Bluetooth® device address

4.6 EPCData

The **EPCData** class defines the output data of the 1800 device.

Namespace: CipherLab.RFID.Device1800.Config
Assembly: CipherLab.RFID.Device1800.Config.dll

Syntax

```
public class EPCData
{
    public DateTime Period { get; set; }
    public string TagData  { get; set; }
    public int TagDataLength { get; set; }
    public byte[] Original { get; set; }
}
```

The EPCData type exposes the following members.

Fields

	Name	Description
public	Original	Unprocessed output data
public	Period	Timestamp of the output data
public	TagData	EPC in hexadecimal value
public	TagDataLength	Length of EPC in words

Index

B

C

D

E

F

H

I

K

L

M

O

R

S

T

U