Device Configuration Specification

Date

01/01/2020

Version 1.0.0

HID Card Reader Configuration

America

20 Fairbanks,
Ste, 170 Irvine, CA 92618
California, U.S.A

Taiwan

114, 3F, No. 21
Ln. 168, Xingshan Rd.
Neihu Dist., Taipei, Taiwan

HID Card Reader Configuration

Device configuration specification

Introduction

For IAdea devices that have HID card reader built-in, users often require to configure the reader to read and output in different formats. This document explains how to configure built-in HID card reader from IAdea device.

Compatible Models

XDS-1088-H
XDS-1588-H

Configuration Steps

1. Create an config.xml file following the configuration format: http://www.a-smil.org/index.php/Device_configuration.
2. Following the specification indicate in next section to add HID reading parameters into config.xml.
3. Copy config.xml to the root DIR of the USB flash drive.
4. Plug USB flash drive to device's USB port. The configuration will be completed after auto-reboot.

Specification

1. NFC Keyboard mode

Enable /disable keyboard mode.

Option	Type	Value
nfc_keyboard_enabled	Boolean	0 = false, 1 = true

2. NFC Keyboard options

Select keyboard options.

Option	Type	Value
nfc_keyboard_options		See 2.1

2.1. NfcKeyboardOptions

Option	Type	Value
onTagDiscovered		See 2.1.1
onTagFormatError	string	Self-input
onTagRemoved	string	Self-input
keyStrokeDownTime	number	Maximun:1000 ms
interKeyStrokeDelay	number	Maximun:1000 ms
upperCaseHexOutput	Boolean	0 = false, 1 = true

- **onTagDiscovered**: Support different statements (see 2.1.1).
- **onTagFormatError**: The input string will display if the format is incorrect.
- **onTagRemoved**: The input string will show when present card then remove the card from reader.
- **keyStrokeDownTime**: The duration (in milliseconds) of each keystroke
- **interKeyStrokeDelay**: The interval (in milliseconds) between consecutive keyboard strokes
- **upperCaseHexOutput**: Show the data in HEX and upper case.

2.1.1. NfcKeyStatement

- **NfcKeyChar**: When presenting the card, it will display single character. Please input the number of ASCII code. (https://www.ascii-code.com/)

  Example:
  onTagDiscovered:{100}.
  Output is “d”.

- **NfcKeyString**: When presenting the card, it will display input string.

  Example:
  onTagDiscovered: "test"
  Output is “test”

- **NfcKeyList**: Display all matching formats data. Separate the data by comma.

  Example:
  onTagDiscovered :{[“A”,”B”]}.
  Output is “AB”

- **NfcKeySelect**: Display first matching formats data.

  Example:
  onTagDiscovered:{select:[{"data":"Tag.id","format":"HEX"},{"data":"Tag.id","format":"BIN"}]}
  Output: 5bba2912ffff12e0

- **NfcKeyCode**: See 2.1.2
- **NfcKeyData**: See 2.1.3

2.1.2. NfcKeyCode

Follow the key event table to input the key event and META_*_ON

Option	Type	Value
KeyCode	string	Key Event Table
MetaState	string	Key Event Table

Example:
onTagDiscovered:{"keyCode":"KEYCODE_A","metaState":"META_CAPS_LOCK_ON"}
Output: “A”

2.1.3. NfcKeyData

Option	Type	Value
data		See 2.1.4
strictBitLength	number	Self-input
customBitOffset	number	Self-input
customBitLength	number	Self-input
leftJustified	boolean	0 = false, 1 = true
unusedBitsLeadingByte	boolean	0 = false, 1 = true
unusedBitsTrailingByte	boolean	0 = false, 1 = true
filteringLeadingBytes	number | string	Self-input
filteringTrailingBytes	number | string	Self-input
reverseBytes	boolean	0 = false, 1 = true
format		See 2.1.4
paddingChar	number | string	Self-input
paddingEnd	boolean	0 = false, 1 = true
paddingLength	number	Self-input

- **data**: See 2.1.4
- **format**: See 2.1.4
- **strictBitLength**: The strict length option specifies the number of bits of data.
- **customBitOffset**: The offset specifies the position in the data, in bits, from which to start outputting data.
- **customBitLength**: The length specifies the number of bytes to read from the card.
  The length needs to be a multiple of 8(bits).
- **leftJustified**:

True: Delete the first 4 bits and additional 4 zeros are added for trailing.

False: Delete the last 4 bits and additional 4 zeros are added for leading.

Example:

onTagDiscovered":{"data":"Tag.id","format":"BIN","customBitOffset":4,"leftJustified":true}

Raw data: 0110 0111 0000 1000 1100 1110 0110

Raw data with leftJustified(True): 0111 0000 1000 1100 1110 0110 0000

- **unusedBitsLeadingByte**: Add 1 byte leading to the raw data received from card.

  Example:

  onTagDiscovered":{"data":"Tag.id","format":"BIN","unusedBitsLeadingByte":true}

  Raw data: 0110 0111 0000 1000 1100 1110 0110

  Raw data with unusedBitsLeadingByte: 0000 0110 0111 0000 1000 1100 1110 0110

- **unusedBitsTrailingByte**: Add 1 byte trailing to the raw data received from card.

  Example:

  onTagDiscovered":{"data":"Tag.id","format":"BIN","unusedBitsTrailingByte":true}

  Raw data: 0110 0111 0000 1000 1100 1110 0110

  Raw data with unusedBitsTrailingByte: 0110 0111 0000 1000 1100 1110 0110 0000

- **filteringLeadingBytes**: The String Filtering function allows specified characters to be removed from the start of raw data.
  Note. The filter character needs to be entered as a decimal coded ASCII value (0~255).

  Example:

  onTagDiscovered:{"data":"Tag.id","format":"HEX","filteringLeadingBytes":91}}

  Raw data: 5bba2912ffff12e0

  Raw data with filteringLeadingBytes: ba2912ffff12e0

- **filteringTrailingBytes**: This function allows specified characters to be removed from the end of raw data.
  Note. The filter character needs to be entered as a decimal coded ASCII value (0~255).

  Example:
  onTagDiscovered:{"data":"Tag.id","format":"HEX","filteringLeadingBytes":224}}

  Raw data: 5bba2912ffff12e0
  Raw data with filteringTrailingBytes: 5bba2912ffff12

- **reverseBytes**: Reverse the standard read order of the card data.

  Example:
  onTagDiscovered:{"data":"Tag.id","format":"HEX","reverseBytes":true
}

  Raw data: 9d11ae3d
  Raw data with Byte Reverse: 3dae119d

- **paddingLength**: The Bit Padding feature allows you to add specified leading bits
  to the raw data received from card.

  Example:
  onTagDiscovered:{"data":"Tag.id","format":"BIN","paddingLength":20}

  Raw data: 0001 0010 0101 1000
  Raw data with Byte paddingEnd: 0001 0010 0101 1000

- **paddingEnd**: If true, this feature allows you to add specified trailing bits to the
  raw data received from card.

  Example:
  onTagDiscovered:{"data":"Tag.id","format":"BIN","paddingEnd":true,"paddingLeng
  th":20}

  Raw data: 0001 0010 0101 1000

Raw data with Byte paddingEnd: 0001 0010 0101 1000 ....

- **paddingChar**: Allows specified character to be added to the input string. Additional
characters can be pasted at the beginning or end of the data.

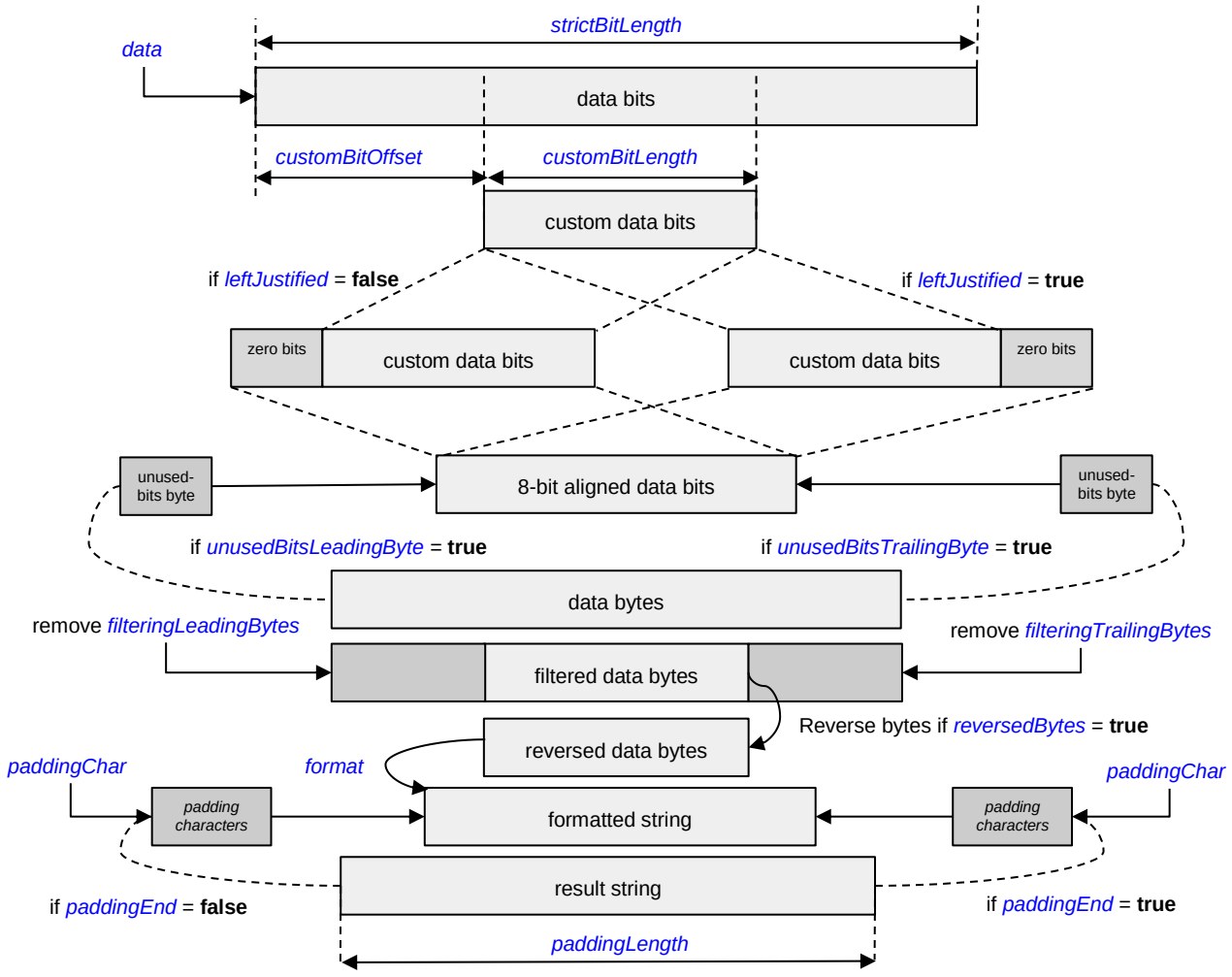
  Example:
  onTagDiscovered:{"data":"Tag.id","format":"BIN","paddingChar":"A", "paddingLength":20}}

  PACS data: 0110 0111 0000 1000
  PACS data with paddingChar: AAAA 0110 0111 0000 1000

2.1.4. NfcDataField & NfcDataFormat

Option	Type	Value
data	string	Tag.id | HidGlobal.pacs
format	string	BIN|HEX|ASCII|BCD|DEC

Data Model



Appendix

Example A

Card type and format: Corporate 1000 – 35-bit Company Code + Card Number ‘Controlled’

```
COMPANY ID CODE,4095,MSB,BIN
CARD ID NUMBER,0,1048575,MSB,BIN
.............PPAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBP
...............111111111111.....................
..............EXX.XX.XX.XX.XX.XX.XX.XX.XX.XX.XX.
..............XX.XX.XX.XX.XX.XX.XX.XX.XX.XX.XX.O
.............OXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
OMNIKEY 5x27 Setup Card Number:
Offset:14
Length:20
```

Sample config.xml for device without password:

```
<configuration xmlns="http://schemas.adfotain.org/config-1.0">
    <userPref>
            <prop name="nfc.keyboard.enabled" value="true"/>
            <prop name="nfc.keyboard.options"
value='{"onTagDiscovered":[{"data": "HidGlobal.pacs","strictBitLength":
35,"customBitOffset": 14,"customBitLength": 20,"format":
"DEC"},{"keyCode":"KEYCODE_ENTER"}]}'/>
    </userPref>
</configuration>
```