8680i
WiFi Software Development Kit

User Guide
Disclaimer

Honeywell International Inc. ("HII") reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult HII to determine whether any such changes have been made. The information in this publication does not represent a commitment on the part of HII.

HII shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material. HII disclaims all responsibility for the selection and use of software and/or hardware to achieve intended results.

This document contains proprietary information that is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated into another language without the prior written consent of HII.

Copyright © 2018 Honeywell International Inc. All rights reserved.

Web Address: www.honeywellaidc.com

For patent information, refer to www.hsmpats.com.


The Bluetooth® word mark and logos are owned by Bluetooth SIG, Inc.

Android™ is a trademark of Google Inc.

Apple is a trademark of Apple Inc., registered in the U.S. and other countries.

Other product names or marks mentioned in this document may be trademarks or registered trademarks of other companies and are the property of their respective owners.
TABLE OF CONTENTS

Customer Support ........................................................................................................................ iii
Technical Assistance ............................................................................................................. iii
Product Service and Repair ................................................................................................ iii
Limited Warranty .................................................................................................................... iii

Chapter 1 - Get Started .................................................................................... 1

About This Manual......................................................................................................................... 1
System Requirements .................................................................................................................. 1
Target Operating Systems ........................................................................................................ 1
Use the SDK................................................................................................................................. 1
  C++ Programming .................................................................................................................. 2
  C# Programming .................................................................................................................... 2
Device Detection............................................................................................................................. 2

Chapter 2 - Definitions..................................................................................... 3

  ButtonPressFlag_WiFi ........................................................................................................... 3
  ButtonPressNotify ................................................................................................................... 3
  DecodeResult_WiFi ................................................................................................................. 3
  ELoggingLevel .......................................................................................................................... 4
  FlashResult ............................................................................................................................. 4
  LanguageOptions_WiFi .......................................................................................................... 4
  MenuCmdResponse ................................................................................................................. 5
  Return Value ............................................................................................................................ 5
  ScannerInfo_WiFi ..................................................................................................................... 5
  ScannerStatus_WiFi ................................................................................................................ 6
### Chapter 3 - API

<table>
<thead>
<tr>
<th>API List</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartServer</td>
<td>10</td>
</tr>
<tr>
<td>StopServer</td>
<td>11</td>
</tr>
<tr>
<td>RegCallbacksWithWifi</td>
<td>11</td>
</tr>
<tr>
<td>UnregResponseCallbackWifi</td>
<td>12</td>
</tr>
<tr>
<td>SetSymbPropWithWifi</td>
<td>12</td>
</tr>
<tr>
<td>GetSymbPropWithWifi</td>
<td>13</td>
</tr>
<tr>
<td>CancelDecodeWifi</td>
<td>14</td>
</tr>
<tr>
<td>DecodeAsyncWithWifi</td>
<td>14</td>
</tr>
<tr>
<td>SetLogLevelWifi</td>
<td>14</td>
</tr>
<tr>
<td>SetDisplayTextWithWifi</td>
<td>15</td>
</tr>
<tr>
<td>SetDisplayColorWithWifi</td>
<td>15</td>
</tr>
<tr>
<td>SetTextSizeWithWifi</td>
<td>16</td>
</tr>
<tr>
<td>EnableNfyBtnPressWithWifi</td>
<td>17</td>
</tr>
<tr>
<td>EnableNfyBtnPressBarcodeWithWifi</td>
<td>17</td>
</tr>
<tr>
<td>SendMenuCmdWithWifi</td>
<td>18</td>
</tr>
<tr>
<td>ShowStatusAlertWithWifi</td>
<td>18</td>
</tr>
<tr>
<td>GetGen7WifiSDKVersion</td>
<td>19</td>
</tr>
<tr>
<td>SetLanguageWithWifi</td>
<td>19</td>
</tr>
<tr>
<td>GetClientAddress</td>
<td>19</td>
</tr>
<tr>
<td>SetDisplayColorHexWithWifi</td>
<td>20</td>
</tr>
<tr>
<td>SetFirmwareInfo</td>
<td>21</td>
</tr>
<tr>
<td>AddFlashingClient</td>
<td>21</td>
</tr>
<tr>
<td>FlashFirmware</td>
<td>21</td>
</tr>
<tr>
<td>OnFileTransfering</td>
<td>22</td>
</tr>
<tr>
<td>OnFileFlashed</td>
<td>22</td>
</tr>
</tbody>
</table>
Chapter 4 - Sample Code .........................................................................................25

Start/Stop Server ...........................................................................................................25
  Start Server .............................................................................................................25
  Stop Server ..........................................................................................................26

Connection/Disconnection ............................................................................................27
  Handle Connection Event .......................................................................................27
  Handle Disconnection Event ..................................................................................27

Configure Scanner .........................................................................................................28
  Pre-Defined Menu Command Parameters ..............................................................28

Trigger a Scan ...............................................................................................................28
  Scan Asynchronously ...............................................................................................28
  Cancel Scan .............................................................................................................28

Send Menu Command ..................................................................................................29

Configure Screen Layout .............................................................................................29
  Set Language ..........................................................................................................29
  Set Display Text ......................................................................................................29
  Set Text Color .........................................................................................................29
  Set Text Size ..........................................................................................................30
  Configure Text Properties for Up and Bottom Lines .............................................30
  Configure Text Properties for Single Line .............................................................30

Show Alert Popup .........................................................................................................30

Get Version ....................................................................................................................31

Handle Button Press Event ..........................................................................................31

Flash Firmware .............................................................................................................32
Customer Support

Technical Assistance

To search our knowledge base for a solution or to log in to the Technical Support portal and report a problem, go to www.hsmcontactsupport.com.

For our latest contact information, see www.honeywellaidc.com/locations.

Product Service and Repair

Honeywell International Inc. provides service for all of its products through service centers throughout the world. To obtain warranty or non-warranty service, return your product to Honeywell (postage paid) with a copy of the dated purchase record. To learn more, go to www.honeywellaidc.com and select Service & Repair at the bottom of the page.

Limited Warranty

For warranty information, go to www.honeywellaidc.com and click Get Resources > Product Warranty.
About This Manual

The 8680i WiFi Software Development Kit (SDK) provides a set of tools and sample source code to help software developers create Windows® desktop applications for the 8680i Wearable Mini-Mobile Computer using Bluetooth SPP protocol.

The following abbreviations are used in this guide:
- API  Application Programming Interface
- SPP  Serial Port Profile

System Requirements

.Net Framework 4.0 must be on the system.

Target Operating Systems

Microsoft® Windows® 7 and Windows 10, 32 and 64 bit.

Use the SDK

There are four folders inside the installation folder:
- include
- lib
- bin
- samples
C++ Programming

• Add the header files HonScannerWifiAPI.h, HonScannerSettings.h and HonScannerWifiStructs.h from the include folder into your application project.

• In the C++ desktop application, link the released lib file Gen7WiFiSDK.lib under the lib folder. Gen7WiFiSDK.lib has different versions for 32bit and 64bit. Make sure the right version is integrated into your application.

• The Gen7WiFiSDK.dll is in the bin folder and the sample projects are in the samples folder.

C# Programming

• Add the Gen7WiFiSDKAssembly.dll from the bin folder into your desktop application project.

Device Detection

Refer to the 8680i User Guide for information about connecting to a WiFi network.
The following definitions are in the `HonScannerWiFiStructs.h` file.

**ButtonPressFlag_WiFi**
- Enumeration.

**Button Press Flag Enumerations**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoButtonPressed_WiFi</td>
<td>No button is pressed</td>
</tr>
<tr>
<td>LeftButtonPressed_WiFi</td>
<td>Left button is pressed</td>
</tr>
<tr>
<td>RightButtonPressed_WiFi</td>
<td>Right button is pressed</td>
</tr>
<tr>
<td>BothButtonsPressed_WiFi</td>
<td>Left and right button are pressed</td>
</tr>
</tbody>
</table>

**ButtonPressNotify**
- Structure.
- Holds the button pressed notification.

**Button Press Notification Structure**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned long connID</td>
<td>The connection ID for the scanner</td>
</tr>
<tr>
<td>ButtonPressFlag_WiFi</td>
<td>Enumeration. Indicates which button is pressed.</td>
</tr>
<tr>
<td>whichButtonPressed</td>
<td></td>
</tr>
</tbody>
</table>

**DecodeResult_WiFi**
- Structure.
• Holds the decoded bar code message.

**Decode Result Structure**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned long connID</td>
<td>The connection ID for the scanner</td>
</tr>
<tr>
<td>char chCodeID</td>
<td>Honeywell Code ID</td>
</tr>
<tr>
<td>char chAimID</td>
<td>AIM ID, the Symbology Identification</td>
</tr>
<tr>
<td>char chAimModifier</td>
<td>AIM Modifier character</td>
</tr>
<tr>
<td>short nLength</td>
<td>The length of the decode data</td>
</tr>
<tr>
<td>char chMessage[2048]</td>
<td>The decode data buffer</td>
</tr>
</tbody>
</table>

**ELoggingLevel**

• Enumeration.

**Logging Level Enumerations**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELL_NONE</td>
<td>No log entries are generated, default value.</td>
</tr>
<tr>
<td>ELL_DATA</td>
<td>Log the device communication data information</td>
</tr>
<tr>
<td>ELL_ERROR</td>
<td>Log errors and all logs at ELL_DATA level.</td>
</tr>
<tr>
<td>ELL_INFO</td>
<td>Log general information and all logs at ELL_ERROR level</td>
</tr>
<tr>
<td>ELL_DEBUG</td>
<td>Log all information</td>
</tr>
</tbody>
</table>

**FlashResult**

• Enumeration.

• Indicates the firmware flash results of the scanner.

**Flash Result Enumerations**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>Flashed firmware successfully</td>
</tr>
<tr>
<td>ConnectFailed</td>
<td>Can’t connect to the scanner</td>
</tr>
<tr>
<td>FileNotFound</td>
<td>The specified firmware file is not found</td>
</tr>
<tr>
<td>FileRefusedByDevice</td>
<td>The firmware file is refused by the scanner</td>
</tr>
<tr>
<td>FileTransferFailed</td>
<td>Failed to transfer the firmware file to the scanner</td>
</tr>
<tr>
<td>FirmwareFlashFailed</td>
<td>Failed to flash the firmware file for the scanner</td>
</tr>
<tr>
<td>FlashResponseTimeout</td>
<td>The scanner doesn’t response in time</td>
</tr>
</tbody>
</table>

**LanguageOptions_WiFi**

• Enumeration.
- Indicates which language characters display on the scanner screen.

**Language Options Enumerations**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loEnglish_WiFi</td>
<td>English</td>
</tr>
<tr>
<td>loCyrillic_WiFi</td>
<td>Cyrillic</td>
</tr>
</tbody>
</table>

**MenuCmdResponse**

- Structure.
- Holds the response of menu command execution.

**Menu Command Response Structure**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned long connID</td>
<td>The connection ID for the scanner</td>
</tr>
<tr>
<td>unsigned char response[512]</td>
<td>The response of the menu command from the scanner</td>
</tr>
<tr>
<td>unsigned long length</td>
<td>The real length of the response</td>
</tr>
</tbody>
</table>

**Return Value**

- Enumeration.
- API function result codes.

**Return Values**

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT_ERR_NOT_INITIALIZED</td>
<td>SDK was not initialized</td>
</tr>
<tr>
<td>RESULT_SUCCESS_WIFI</td>
<td>Operation was successful</td>
</tr>
<tr>
<td>RESULT_ERR_SERVICE_NOT_STARTED</td>
<td>The TCP socket service was not started</td>
</tr>
<tr>
<td>RESULT_ERR_SERVICE_ALREADY_STARTED</td>
<td>The TCP socket service was already started</td>
</tr>
<tr>
<td>RESULT_ERR_PARAMETER_WIFI</td>
<td>One of the function parameters was invalid</td>
</tr>
<tr>
<td>RESULT_ERR_UNSUPPORTED_WIFI</td>
<td>The operation was not supported by the engine</td>
</tr>
<tr>
<td>RESULT_ERR_EXCEPTION_WIFI</td>
<td>An exception was detected in the engine</td>
</tr>
</tbody>
</table>

**ScannerInfo_WiFi**

- Structure.
- Holds the scanner information.

**Scanner Information Structure**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned long connID</td>
<td>The connection ID for the scanner</td>
</tr>
</tbody>
</table>
Scanner Information Structure (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char chName[128]</td>
<td>The module name of the scanner</td>
</tr>
<tr>
<td>char chDesc[128]</td>
<td>The description of the scanner</td>
</tr>
<tr>
<td>char chSerialNum[128]</td>
<td>The serial number of the scanner</td>
</tr>
<tr>
<td>char chAppVersion[128]</td>
<td>The application version of the scanner</td>
</tr>
<tr>
<td>char chAppDate[64]</td>
<td>The application date of the scanner</td>
</tr>
<tr>
<td>char chAppTime[64]</td>
<td>The application time of the scanner</td>
</tr>
<tr>
<td>char chBootVersion[128]</td>
<td>The boot version of the scanner</td>
</tr>
<tr>
<td>char chBootDate[64]</td>
<td>The boot date of the scanner</td>
</tr>
<tr>
<td>char chBootTime[64]</td>
<td>The boot time of the scanner</td>
</tr>
</tbody>
</table>

ScannerStatus_WiFi

- Enumeration.
- Sets alert popup.

Scanner Status Enumerations

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssNormal_WiFi</td>
<td>Don’t show any popup</td>
</tr>
<tr>
<td>ssBadScan_WiFi</td>
<td>Show bad scan alert</td>
</tr>
<tr>
<td>ssGoodScan_WiFi</td>
<td>Show good scan popup</td>
</tr>
</tbody>
</table>

SymbPropResponse

- Structure.
- Holds the symbology property values.

Symbology Property Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned long connID</td>
<td>The connection ID for the scanner</td>
</tr>
<tr>
<td>unsigned long propID</td>
<td>The ID for the symbology property</td>
</tr>
<tr>
<td>unsigned long value</td>
<td>The value for the specified symbology property</td>
</tr>
<tr>
<td>bool successful</td>
<td>Indicates whether the operation is successful</td>
</tr>
</tbody>
</table>

TextColors_WiFi

- Enumeration.
• Sets foreground or background text color.

**TextColor Enumerations**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultColor_WiFi</td>
<td>Background color, default is black.</td>
</tr>
<tr>
<td></td>
<td>Foreground color, default is white.</td>
</tr>
<tr>
<td>Red_WiFi</td>
<td>Red color</td>
</tr>
<tr>
<td>Green_WiFi</td>
<td>Green color</td>
</tr>
<tr>
<td>Blue_WiFi</td>
<td>Blue color</td>
</tr>
</tbody>
</table>

**TextColorType_WiFi**

• Enumeration.

• Sets foreground or background color on the display.

**TextColor Type Enumerations**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BgColor_WiFi</td>
<td>Background color</td>
</tr>
<tr>
<td>FgColorUpLine_WiFi</td>
<td>Foreground color for up line</td>
</tr>
<tr>
<td>FgColorBottomLine_WiFi</td>
<td>Foreground color for bottom line</td>
</tr>
</tbody>
</table>

**TextFontSizes_WiFi**

• Enumeration.

• Sets text font size.

**Text Font Size Enumerations**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small_WiFi</td>
<td>Small size</td>
</tr>
<tr>
<td>Medium_WiFi</td>
<td>Medium size, default value.</td>
</tr>
<tr>
<td>Large_WiFi</td>
<td>Large size</td>
</tr>
<tr>
<td>ExtraLarge_WiFi</td>
<td>Extra large size</td>
</tr>
<tr>
<td>SmallBold_WiFi</td>
<td>Small and bold</td>
</tr>
<tr>
<td>MediumBold_WiFi</td>
<td>Medium and bold</td>
</tr>
<tr>
<td>LargeBold_WiFi</td>
<td>Large and bold</td>
</tr>
<tr>
<td>ExtraLargeBold_WiFi</td>
<td>Extra large and bold</td>
</tr>
<tr>
<td>SingleLineSmall_WiFi</td>
<td>Small size for single line, only works for the up line</td>
</tr>
<tr>
<td>SingleLineMedium_WiFi</td>
<td>Medium size for single line, only works for the up line</td>
</tr>
<tr>
<td>SingleLineLarge_WiFi</td>
<td>Large size for single line, only works for the up line</td>
</tr>
</tbody>
</table>
**TextLineType_WiFi**

- Enumeration.
- Sets on which line text should be set on the display.

**Text Line Type Enumerations**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpLine_WiFi</td>
<td>Up line</td>
</tr>
<tr>
<td>BottomLine_WiFi</td>
<td>Bottom line</td>
</tr>
</tbody>
</table>
API List

Windows Native C/C++ APIs are listed below. They have the same return value as defined in Return Value (page 5). Additional information about APIs is available in the HonScannerWiFiAPI.h file.

### C/C++ API List

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartServer</td>
<td>Starts the TCP socket server for incoming scanner connections.</td>
</tr>
<tr>
<td>StopServer</td>
<td>Stops the TCP socket server and releases the connections.</td>
</tr>
<tr>
<td>RegCallbacksWithWifi</td>
<td>Register the Callbacks in the SDK. SDK forwards the events to them.</td>
</tr>
<tr>
<td>UnregResponseCallbackWifi</td>
<td>Stop receiving events from the SDK layer.</td>
</tr>
<tr>
<td>SetSymbPropWithWifi</td>
<td>Set the symbology settings of the scanner.</td>
</tr>
<tr>
<td>GetSymbPropWithWifi</td>
<td>Get the symbology settings from the scanner.</td>
</tr>
<tr>
<td>CancelDecodeWifi</td>
<td>Cancel decode action in process.</td>
</tr>
<tr>
<td>DecodeAsyncWithWifi</td>
<td>Trigger an asynchronous scan and don’t wait for the result.</td>
</tr>
<tr>
<td>SetLogLevelWifi</td>
<td>Set the log level for outputting the specified log information.</td>
</tr>
<tr>
<td>SetDisplayTextWithWifi</td>
<td>Set the text content to display on the scanner screen through sending a menu command asynchronously.</td>
</tr>
<tr>
<td>SetDisplayColorWithWifi</td>
<td>Set the background or foreground color of the text on the scanner screen through sending a menu command asynchronously.</td>
</tr>
<tr>
<td>SetTextSizeWithWifi</td>
<td>Set the text font size of the text on the scanner screen through sending a menu command asynchronously.</td>
</tr>
<tr>
<td>EnableNfyBtnPressWithWifi</td>
<td>Make the scanner send a notification to the host when one or more scanner buttons are pressed.</td>
</tr>
</tbody>
</table>
StartServer

Starts the TCP socket server for incoming scanner connections.

**Parameters**

- **wchar_t** `lpszBindAddress`
  The server IP address.

- **unsigned long** `usPort`
  The server port number.

**Return Value**

RESULT_SUCCESS_WIFI if the TCP socket server is successfully started.
RESULT_ERR_PARAMETER_WIFI if the lpszBindAddress is null or usPort is out of range (1 - 65535).
RESULT_ERR_SERVICE_ALREDAY_STARTED if the TCP server has already started.
RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_EXCEPTION_WIFI if the server is not successfully started.

StopServer

Stops the TCP socket server and releases the connections.

**Note:** This API waits until all jobs are done then returns. It must be called in a thread because this may take some time and can block the main thread.

**Parameters**

N.A.

**Return Value**

RESULT_SUCCESS_WIFI if the TCP server is closed successfully.
RESULT_ERR_SERVICE_NOT_STARTED if the TCP server is not started.
RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_EXCEPTION_WIFI if the server is not successfully stopped.

RegCallbacksWithWifi

This API allows the application to register a callback to receive the events from the SDK such as bar code responses, disconnect events, button press events, and other responses from the scanner.

**Parameters**

**OnConnectCallback connCb**

The function pointer that receives the connect events from the SDK layer.

```c
typedef void (*OnConnectCallback)(const ScannerInfo_WiFi &info);
```

**OnDisconnectCallback disconnCb**

The function pointer that receives disconnect events from the SDK layer.

```c
typedef void (*OnDisconnectCallback)(const CLIENT_CONNID connID);
```

**OnDecodeCallback decCb**
The function pointer that receives scan bar code events from the SDK layer.

```c
typedef void (*OnDecodeCallback)(const DecodeResult_WiFi &decRes);
```

**OnPressButtonCallback pressBtnCb**

The function pointer that receives button press events from the SDK layer.

```c
typedef void (*OnPressButtonCallback)(const ButtonPressNotify &notify);
```

**OnGetSymbPropCallback symbPropCb**

The function pointer that receives get/set symbology property events from the SDK layer.

```c
typedef void (*OnGetSymbPropCallback)(const SymbPropResponse &resp);
```

**OnSendMenuCmdCallback menuCmdCb**

The function pointer that receives the menu command responses from the SDK layer.

```c
typedef void (*OnSendMenuCmdCallback)(const MenuCmdResponse &resp);
```

**Return Value**

- RESULT_SUCCESS_WIFI if successfully registered.
- RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.

**UnregResponseCallback Wifi**

This is used to stop receiving events from the SDK layer.

**Parameters**

N.A.

**Return Value**

- RESULT_SUCCESS_WIFI if successfully unregistered.
- RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.

**SetSymbPropWithWifi**

Set the symbol code property in the scanner by sending an asynchronous menu command.
Parameters

unsigned long connID
The connection ID for the scanner.

unsigned long symbolCodeID
The symbol code properties (found in the HonScannerSettings.h file).

unsigned long value
The new value for the property to set.

Return Value

RESULT_SUCCESS_WIFI if the menu command is successfully sent.
RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_SERVICE_NOT_STARTED if the TCP server is not started.
RESULT_ERR_PARAMETER_WIFI if the connID or symbolCodeID are not correct.
RESULT_ERR_EXCEPTION_WIFI if the menu command fails.

GetSymbPropWithWifi

Get the property of the symbol code from the scanner via sending menu command asynchronously.

Parameters

unsigned long connID
The connection ID for the scanner.

Unsigned long symbolCodeID
The symbol code properties (found in the HonScannerSettings.h file).

Return Value

RESULT_SUCCESS_WIFI if the property is successfully retrieved.
RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_SERVICE_NOT_STARTED if the TCP server is not started.
RESULT_ERR_PARAMETER_WIFI if the connID or symbolCodeID are not correct.
RESULT_ERR_EXCEPTION_WIFI if the menu command fails.
**CancelDecodeWifi**

Cancel the decode action in process.

**Parameters**

*unsigned long connID*

The connection ID for the scanner.

**Return Value**

RESULT_SUCCESS_WIFI if the decode is successfully canceled.
RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_SERVICE_NOT_STARTED if the TCP server is not started.
RESULT_ERR_PARAMETER_WIFI if the connID is not correct.
RESULT_ERR_EXCEPTION_WIFI if the menu command for canceling the decode fails.

**DecodeAsyncWithWifi**

Trigger an asynchronous scan and return the scan result with a callback event.

**Parameters**

*unsigned long connID*

The connection ID for the scanner.

**Return Value**

RESULT_SUCCESS_WIFI if the decode menu command is successfully canceled.
RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_SERVICE_NOT_STARTED if the TCP server is not started.
RESULT_ERR_PARAMETER_WIFI if the connID is not correct.
RESULT_ERR_EXCEPTION_WIFI if the menu command for scanning fails.

**SetLogLevelWifi**

Set the log level for the specified log entries output.

**Parameters**

*EloggingLevel*
Indicates the log level. See `ELoggingLevel` for more details.

**Return Value**

N.A.

*This API doesn’t rely on the connection to scanner, so you can invoke it before invoking* StartServer.

### SetDisplayTextWithWifi

Set the text content to display on the scanner screen by sending an asynchronous menu command.

**Parameters**

- **unsigned long connID**
  The connection ID for the scanner.

- **TextLineType_WiFi whichLine**
  Enumerations: UpLine_WiFi, BottomLine_WiFi. See `TextLineType_WiFi` for more details.

- **const wchar_t* text**
  The text to display. Supports Unicode.

**Return Value**

- `RESULT_SUCCESS_WIFI` if the menu command for setting display text is successfully sent.
- `RESULT_ERR_NOTInicialize` if the SDK is not successfully initialized.
- `RESULT_ERR_SERVICE_NOT_STARTED` if the TCP server is not started.
- `RESULT_ERR_PARAMETER_WIFI` if the `connID` is not correct or text is null.
- `RESULT_ERR_EXCEPTION_WIFI` if the menu command for setting display text fails.

### SetDisplayColorWithWifi

Set the background or foreground color of the text on the scanner screen by sending an asynchronous menu command.

**Parameters**

- **unsigned long connID**
  The connection ID for the scanner.
TextColorType_WiFi colorType
Enumerations: BgColor_WiFi, FgColorUpLine_WiFi, FgColorBottomLine_WiFi. See TextColorType_WiFi for more details.

TextColors_WiFi color
Enumerations: DefaultColor_WiFi, Red_WiFi, Green_WiFi, Blue_WiFi. See TextColors_WiFi for more details.

Return Value
RESULT_SUCCESS_WIFI if the menu command for setting text color is successfully sent.
RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_SERVICE_NOT_STARTED if the TCP server is not started.
RESULT_ERR_PARAMETER_WIFI if the connID is not correct.
RESULT_ERR_EXCEPTION_WIFI if the menu command for setting text color fails.

SetTextSizeWithWifi
Set the text font size on the scanner screen by sending an asynchronous menu command.

Parameters
unsigned long connID
The connection ID for the scanner.

TextLineType_WiFi whichLine
Enumerations: UpLine_WiFi, BottomLine_WiFi. See TextLineType_WiFi for more details.

TextFontSizes_WiFi fontSize
Enumerations: Small_WiFi, Medium_WiFi, Large_WiFi. See TextFontSizes_WiFi for more details.

Return Value
RESULT_SUCCESS_WIFI if the menu command for setting font size is successfully sent.
RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_SERVICE_NOT_STARTED if the TCP server is not started.
RESULT_ERR_PARAMETER_WIFI if the connID is not correct.
RESULT_ERR_EXCEPTION_WIFI if the menu command for setting font size fails.
EnableNfyBtnPressWithWifi

Make the scanner send a notification to the host when one or both of the scanner buttons are pressed. This is done by sending an asynchronous menu command.

Parameters

- **unsigned long connID**
  The connection ID for the scanner.

- **bool enable**
  True or false.

Return Value

- **RESULT_SUCCESS_WIFI** if the command for the enable/disable button press is successfully sent.
- **RESULT_ERR_NOT_INITIALIZE** if the SDK is not successfully initialized.
- **RESULT_ERR_SERVICE_NOT_STARTED** if the TCP server is not started.
- **RESULT_ERR_PARAMETER_WIFI** if the `connID` is not correct.
- **RESULT_ERR_EXCEPTION_WIFI** if the menu command for enable/disable button press notification fails.

*Note:* If you want to receive the button pressed callback event, you should invoke this API when the scanner is connected, for example, in the connected callback function.

EnableNfyBtnPressBarcodeWithWifi

Make the scanner send a notification to the host when one or both of the scanner buttons are pressed and bar code data is sent. This is done by sending an asynchronous menu command.

Parameters

- **unsigned long connID**
  The connection ID for the scanner.

Return Value

- **RESULT_SUCCESS_WIFI** if the command for enabling notifications is successfully sent.
- **RESULT_ERR_NOT_INITIALIZE** if the SDK is not successfully initialized.
- **RESULT_ERR_SERVICE_NOT_STARTED** if the TCP server is not started.
- **RESULT_ERR_PARAMETER_WIFI** if the `connID` is not correct.
RESULT_ERR_EXCEPTION_WIFI if the menu command enabling/disabling button press notifications and bar code data transmission fails.

**SendMenuCmdWithWifi**

Send the raw menu command to the scanner asynchronously.

**Parameters**

- **unsigned long connID**
  The connection ID for the scanner.
- **const char* cmd**
  The raw text of the menu command.

Add the command prefix such as SYN_M or SYN_Y and the command suffix such as RAM(!) or ROM(.).

This function can send a series of commands with a separator (;), such as EA8ENA1;C39ENA1;128ENA1. The length of menu command response is limited to 512 so don’t make the menu command too long.

**Return Value**

- RESULT_SUCCESS_WIFI if the menu command is successfully executed.
- RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.
- RESULT_ERR_SERVICE_NOT_STARTED if the TCP server is not started.
- RESULT_ERR_PARAMETER_WIFI if the **connID** is not correct, or the **cmd** is null or empty.
- RESULT_ERR_EXCEPTION_WIFI if the menu command fails.

**ShowStatusAlertWithWifi**

Show the status image on the scanner screen by an asynchronous menu command.

**Parameters**

- **unsigned long connID**
  The connection ID for the scanner.
- **ScannerStatus_WiFi status**
  The scanner status, such as good scan or bad scan. See **ScannerStatus_WiFi**.
GetGen7WifiSDKVersion

Get the current version of the SDK.

Parameters

char* version
The array for receiving the version.

int* verSize
[In] The max size of the returned version array to pass in.
[Out] The real size of the returned version array, not larger than the max size passed in.

Return Value
RESULT_SUCCESS_WIFI if the SDK version is successfully retrieved.
RESULT_ERR_PARAMETER_WIFI if the version or verSize is null.
RESULT_ERR_EXCEPTION_WIFI if the version retrieval fails.

SetLanguageWithWifi

Parameters

unsigned long connID
The connection ID for the scanner.

LanguageOptions_WiFi option
The language options. See LanguageOptions_WiFi.

Return Value
RESULT_SUCCESS_WIFI if the menu command is successfully executed.
RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_SERVICE_NOT_STARTED if the TCP server is not started.
RESULT_ERR_PARAMETER_WIFI if the connID is not correct.
RESULT_ERR_EXCEPTION_WIFI if the menu command fails.

GetClientAddress

Get the IP address and port of the socket of the connected scanner.
Parameters

unsigned long connID
The connection ID for the scanner.

wchar_t* lpszAddress
The array for receiving the IP address.

int& iAddrLen
The returned length of IP address.

unsigned short& usPort
The returned port number.

Return Value
RESULT_SUCCESS_WIFI if the IP address and port are successfully retrieved.
RESULT_ERR_NOT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_SERVICE_NOT_STARTED if the TCP server is not started.
RESULT_ERR_PARAMETER_WIFI if the connID is not correct or the lpszAddress is null.
RESULT_ERR_EXCEPTION_WIFI if the IP address and port retrieval fails.

SetDisplayColorHexWithWifi

Set the background or foreground color of the text on the scanner screen by sending an asynchronous menu command.

Parameters

unsigned long connID
The connection ID for the scanner.

TextColorType_WiFi colorType
Enumerations: BgColor_WiFi, FgColorUpLine_WiFi, FgColorBottomLine_WiFi. See TextColorType_WiFi for more details.

const char* hexColor
The RGB hex code string.

Return Value
RESULT_SUCCESS_WIFI if the menu command for setting text color is successfully sent.
RESULT_ERR_NOT_INITIALIZED if the SDK is not successfully initialized.
RESULT_ERR_SERVICE_NOT_STARTED if the TCP server is not started.
RESULT_ERR_PARAMETER_WIFI if the `connID` is not correct, if `hexColor` is null, or its length is not 6.
RESULT_ERR_EXCEPTION_WIFI if the menu command for setting text color fails.

**SetFirmwareInfo**

*Note:* This API is only for C# version assembly.

This sets the full path and version of the firmware file.

**Parameters**

- **string filePath**
  The full path of the firmware file.

- **string version**
  The version of the firmware file.

**Return Value**

N.A.

**AddFlashingClient**

*Note:* This API is only for C# version assembly.

Add the scanner to be flashed.

**Parameters**

- **uint connID**
  The connection ID for the scanner.

**Return Value**

N.A.

**FlashFirmware**

*Note:* This API is only for C# version assembly.
Flash the firmware file to the scanners. The firmware file information is set by `SetFirmwareInfo` and the scanners are added by `AddFlashingClient`. To reduce the network transfer load, schedule 10 threads in a pool to transfer a firmware file to scanners concurrently.

**Parameters**

N.A.

**Return Value**

N.A.

**OnFileTransfering**

*Note: This API is only for C# version assembly.*

This delegate function is invoked when sending a firmware file to the scanner.

```csharp
public delegate void OnFileTransfering(ulong connID, int sentSize, int totalSize);
```

**Parameters**

- **uint connID**
The connection ID for the scanner.
- **int sentSize**
The size already sent to the scanner.
- **int totalSize**
The total size of the firmware file.

**Return Value**

N.A.

**OnFileFlashed**

*Note: This API is only for C# version assembly.*

This delegate function is invoked when the flash is complete or an exception occurs.

```csharp
public delegate void OnFileFlashed(ulong connID, FlashResult res);
```

**Parameters**

- **uint connID**
The connection ID for the scanner.

**FlashResult res**
The result of flashing. See `FlashResult` for more details.

**Return Value**
N.A.
Start/Stop Server

Start Server

// Logging doesn’t rely on the Server, so we can initialize the level at the beginning.
SetLogLevelWifi(ELL_DEBUG);

Result_t res = StartServer(strIP, lPort);
If (res == RESULT_SUCCESS)
{
    // Register the callback functions for receiving the events.
    RegCallbacksWithWifi(OnConnect, OnDisconnect,
                        OnDecode, OnPressButton,
                        OnGetSymbProp, OnSendMenuCmd);
}
Stop Server

The following CStopServerThread and CApp are the classes of the Desktop application.

// The thread is working for stopping server which invokes the StopServer API really.
typedef void (*OnStopServerCallback)();

// The callback is invoked to tell the main thread that the stopping server is done.
void CStopServerThread::setStopServerCallback(OnStopServerCallback stopSrvCb)
{
    m_pStopSrvCb = stopSrvCb;
}

void CStopServerThread::stopServer()
{
    stopEvent.signal();
}

void CStopServerThread::entry()
{
    while(stopEvent.wait())
    {
        StopServer();
        if(m_pStopSrvCb != nullptr)
            m_pStopSrvCb();
        stopEvent.unsignal();
    }
}

// The main thread of the application to stop server.
CStopServerThread m_stopSrvThread;

void CApp::StopServer()
{
    m_stopSrvThread.setStopServerCallback(OnStopServer);
    m_stopSrvThread.stopServer(); // Stop the server in another thread.
}

// Stop server callback for the thread
void OnStopServer()
{

Connection/Disconnection

Handle Connection Event

// Store the scanner information when it connects to the server in the callback
// function
void OnConnect(const ScannerInfo_WiFi &info)
{
    // Note: The info passed from callback may be released by SDK,
    // so we should deep copy it and store locally.
    ScannerInfo_WiFi scanner;
    memcpy(&scanner, &info, sizeof(ScannerInfo_WiFi));
    m_vcScanners.push(scanner);

    // Enable to receive button pressed callback events once connect to the
    // scanner. Otherwise you can invoke this API wherever as you need.
    EnableNfyBtnPressWithWifi(scanner.ConnID, true);
}

Handle Disconnection Event

// Remove the scanner information when it disconnects to the server in the
// callback function
void OnDisconnect(const unsigned long connID)
{
    for(ScannerIT it = m_vcScanners.begin(); it != m_vcScanners.end(); ++it)
    {
        if((*it)->connID == scannerID)
        {
            it = m_vcScanners.erase(it);
            break;
        }
    }
}
Configure Scanner

Pre-Defined Menu Command Parameters

```cpp
SetSymbPropWithWifi(connID, DEC_EAN8_ENABLED, 1); // Enable EAN-8
SetSymbPropWithWifi(connID, DEC_EAN8_CHECK_DIGIT_TRANSMIT, 1);
SetSymbPropWithWifi(connID, DEC_EAN8_2CHAR_ADDENDA_ENABLED, 0); // Disable 2 char addenda
SetSymbPropWithWifi(connID, DEC_CODE128_MIN_LENGTH, 5); // Set the minimum length of code 128 to be 5

// Retrieve the symbology property result in the callback function
void OnGetSymbProp(const SymbPropResponse &resp)
{
    if(resp.successful)
    {
        OutputSymbProp(resp);
    }
}
```

Trigger a Scan

Scan Asynchronously

```cpp
DecodeAsyncWithWifi(connID); // Send scan command

// Retrieve the scan result in the callback function
void OnDecode(const DecodeResult_WiFi &decRes)
{
    OutputDecodeResult(decRes);
}
```

Cancel Scan

```cpp
CancelDecodeWifi(connID); // Cancel scan
```
**Send Menu Command**

```c
#define CMD_SYN_M  "\x16\x4d\x0d"
#define CMD_RAM    "\x21"  // !

std::string cmd = "BT_NAM?";
std::string cmd = CMD_SYN_M + cmd + CMD_RAM;
SendMenuCmdWithWifi(connID, cmd.c_str());

// Retrieve the menu command result in the callback function
void OnSendMenuCmd(const MenuCmdResponse &resp)
{
    std::string str((char*)resp.response, resp.length);
    OutputMenuCmdResponse(str);
}
```

**Configure Screen Layout**

**Set Language**

```c
SetLanguageWithWifi(connID, loCyrillic_WiFi); // Set the scanner to be ready to show Cyrillic
```

**Set Display Text**

```c
setDisplayTextWithWifi(connID, UpLine_WiFi, "Welcome");  // Show ‘Welcome’ at the up line
setDisplayTextWithWifi(connID, BottomLine_WiFi, "Bad Code");  // Show ‘Bad Code’ at the bottom line
```

**Set Text Color**

```c
// Show the background color in Red
setDisplayColorWithWifi(connID, BgColor_WiFi, Red_WiFi);
// Show the foreground color of the up line in Green
setDisplayColorWithWifi(connID, FgColorUpLine_WiFi, Green_WiFi);

// Show the background color in yellow
setDisplayColorHexWithWifi(connID, BgColor_WiFi, "ffff00");
// Show the foreground color of the up line in brown
setDisplayColorHexWithWifi(connID, FgColorUpLine_WiFi, "800000");
```
Set Text Size

// Show the text in Large size at the up line
SetTextSizeWithWifi(connID, UpLine_WiFi, Large_WiFi);
// Show the text in Small size at the bottom line
SetTextSizeWithWifi(connID, BottomLine_WiFi, Small_WiFi);

Configure Text Properties for Up and Bottom Lines

// Show ‘Welcome’ in large size with green foreground and red background at the
// up line
setDisplayColorWithWifi(connID, BgColor_WiFi, Red_WiFi);
setDisplayColorWithWifi(connID, FgColorUpLine_WiFi, Green_WiFi);
SetTextSizeWithWifi(connID, UpLine_WiFi, Large_WiFi);
setDisplayTextWithWifi(connID, UpLine_WiFi, “Welcome”);

// Show ‘Bad Code’ in small size with blue foreground at the bottom line
setDisplayColorWithWifi(connID, FgColorBottomLine_WiFi, Blue_WiFi);
SetTextSizeWithWifi(connID, BottomLine_WiFi, Small_WiFi);
setDisplayTextWithWifi(connID, BottomLine_WiFi, “Bad Code”);

Configure Text Properties for Single Line

// Clear the screen to make sure the old content will not overlap with the new
// content.
setDisplayTextWithWifi(connID, UpLine_WiFi, “”);
setDisplayTextWithWifi(connID, BottomLine_WiFi, “”);

// Show ‘Welcome’ in large size with green foreground and red background at the
// single line.
// Notes: When using SingleLineSmall_WiFi, SingleLineMedium_WiFi,
// SingleLineLarge_WiFi for font sizes, the SetDisplayTextWithWifi should pass
// UpLine_WiFi for displaying content.
setDisplayColorWithWifi(connID, BgColor_WiFi, Red_WiFi);
setDisplayColorWithWifi(connID, FgColorUpLine_WiFi, Green_WiFi);
SetTextSizeWithWifi(connID, UpLine_WiFi, SingleLineLarge_WiFi);
setDisplayTextWithWifi(connID, UpLine_WiFi, “Welcome”);

Show Alert Popup

ShowStatusAlertWithWifi(connID, ssGoodScan_WiFi);
Get Version

```c
char version[20];
int verLen = 20;
Result_t res = GetGen7WifiSDKVersion(version, &verLen);
if(res == RESULT_SUCCESS)
    Log(CString(version, verLen));
```

Handle Button Press Event

```c
// Should enable this functionality first, so we can receive button pressed
// events.
EnableNfyBtnPressWithWifi(connID, true);
// Handle the button press event in the callback function
void OnPressButton(const ButtonPressNotify &notify)
{
    switch(notify.whichButtonPressed)
    {
    case LeftButtonPressed_WiFi:
        // Do something when left button is pressed
        break;
    case RightButtonPressed_WiFi:
        // Do something when right button is pressed
        break;
    }
}
```
Flash Firmware

**Note:** *This API is only for C# version assembly.*

```csharp
// The definitions of the delegate functions for flashing
private void OnFwFileTransfering(ulong connID, int sentSize, int totalSize)
{
    Log(string.Format("Scanner [{0}], Bytes transferred:{1}/{2}", connID,
                      sentSize, totalSize));

    if (sentSize == totalSize)
        Log(string.Format("Scanner [{0}] is flashing firmware. Please wait...",
                           connID));
}

private void OnFwFlashed(ulong connID, FlashResult res)
{
    Log(string.Format("Scanner [{0}], Flash result: {1}", connID,
}

// Register the delegate functions to API assembly at the beginning.
// apiWrapper is the instance of our WiFi SDK assembly.
public void Init()
{
    apiWrapper.OnTransfering = new OnFileTransfering(OnFwFileTransfering);
    apiWrapper.OnFlashed = new OnFileFlashed(OnFwFlashed);
}

// Flash firmware to the selected scanners
private void btnFlashFW_Click(object sender, EventArgs e)
{
    List<uint> ids = GetSelectedScannerIDs();
    if(ids.count == 0)
        return;
    try
    {
        apiWrapper.SetFirmwareInfo(tbFwPath.Text, tbFwVersion.Text);
        foreach (uint id in ids)
        {
            Log(string.Format("Start to flash firmware, connecting to Scanner
                               [{0}] ...", id));

            apiWrapper.AddFlashingClient(id);
        }
    }
```
}  
    apiWrapper.FlashFirmware();
}  
catch (Exception ex)  
{
    Log(string.Format("Exception: {0}", ex.Message));
}
}