

iPEARL S5 (and S4.5)

API description for Communication DLL with NFC Interface

Version 1.0



1 Table of Contents

2	Revision Sheet:	3
3	Introduction	4
3.1	Purpose of this document	4
3.2	Preconditions	4
3.3	Target Audience	4
4	NFC S5 DLL API for communicating with iPEARL	6
4.1	public OpenConnection()	6
4.2	public bool ConnectReader ()	6
4.3	public bool ConnectDevice ()	7
	public bool ConnectDevice()	7
4.4	public bool MCI_Read()	7
4.5	public bool MCI_Write()	8
4.6	public string GetDllVersion()	8
4.7	public bool ST25DV_ReadMultipleBlocks()	9
5	Error Handling	9
6	Structures and commands for iPEARL	10

2 Revision Sheet:

Ver- sion	Date	Author	Description of Change
1.0	16-02-2021	Kailas Gijare	Creation

3 Introduction

3.1 Purpose of this document

This document describes the communication interface with the NFCS5 DLL.

Below is a collection of public method and properties that can be used in a higher application level.

3.2 Preconditions

In order to get full access to all functions of this DLL, following components are required:

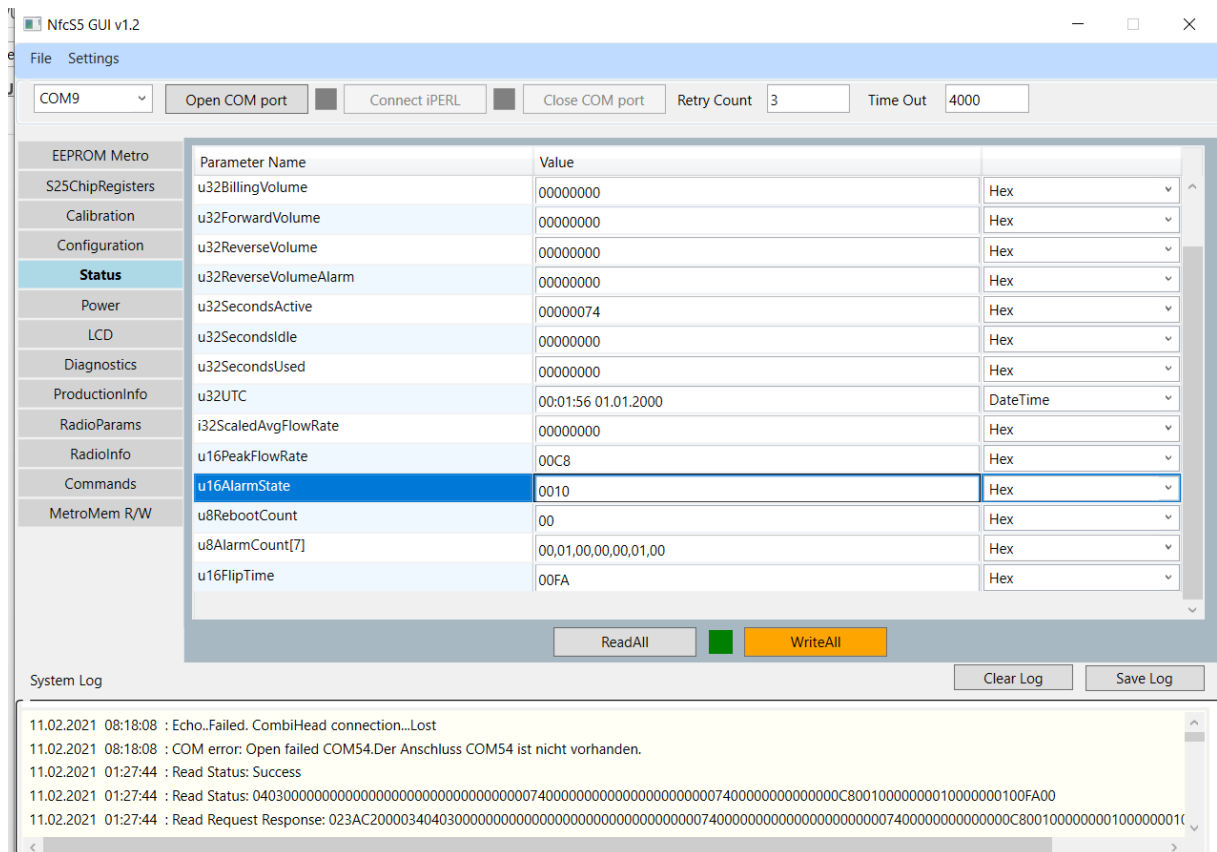
- Communication DLL (NFCS5_DLL.dll)
- Combi-Head to read /Write NFC with CR95 chip
- USB Cable Type A to Type Mini B
- iPEARL S4.5 or above

In order to use this DLL in own applications, the .Net Framework Version 4.5.2 is at least required.

This version can be found here <https://www.microsoft.com/de-de/download/details.aspx?id=42642>

3.3 Target Audience

This document is for developers that want to integrate this DLL in their own application.



This tool is designed for developers, which are familiar with the project scope and have experience with the command setting for the communication with the Cr95Hf Controller.

4 NFCS5 DLL API for communicating with iPEARL

For anyone who wishes to interact with iPEARL equipped with NFC, following list of functions provides an easy interface with which Data Structures and commands can be read/written.

Before writing /reading, any data from iPEARL over NFC, following functions should be called with correct parameters as described to setup NFC and comports:

- `OpenConnection()`
- `ConnectReader()`
- `ConnectDevice()`

After this, individual structures, or chunks of memory, or Commands can be exchanged using `MCI_Write()` or `MCI_Read()` functions.

`GetDllVersion()` and `ST25DV_ReadMultipleBlocks()` are miscellaneous functions provided to read DLL version and Read EE-PROM blocks of ST25 device.

4.1 `public OpenConnection(string comport, int baudrate, int databits, Parity parity, StopBits stopbits)`

Name:	<code>public bool OpenConnection(string comport, int baudrate, int databits, Parity parity, StopBits stopbits)</code>
Purpose	Opens a connection with selected COM port.
Default Parameter Values :	BaudRate: 57600 Data-Bits : 8 Parity : "None" StopBits : "Two"
Return :	True when Successful, False when Error.

4.2 `public bool ConnectReader (void)`

Name:	<code>public bool ConnectReader (void)</code>
Purpose	Connects a CR95XX device for performing NFC operations
Return :	True when Successful, False when Error.

4.3 public bool ConnectDevice (void)

Name:	public bool ConnectDevice(void)
Purpose	Connects a ST25XX device for performing NFC operations
Return :	True when Successful, False when Error.

4.4 public bool MCI_Read(MCI_Protocol.StructName StName, ushort offset, byte payloadlength, int timeoutMs, out byte[] payload, out byte errorCode)

Name:	bool MCI_Read(MCI_Protocol.StructName StName, ushort offset, byte payloadlength, int timeoutMs, out byte[] payload, out byte errorCode)
Purpose	Used to Read Structures/ Memory from device via NFC interface
Return :	True when Successful, False when Error.
Input parameters:	MCI_Protocol.StructName StName: Name of structure to be read or command to be executed. See here for more details Offset : Offset from a Structure or Memory Location PayloadLength: Length of payload to be read TimeoutMs : Timeout in MilliSeconds (default 4000ms)
Output:	- Data is stored in out byte[] payload for further processing. Error code is stored in out byte error code. 0x00 for Success, non-zero value indicates failure

4.5 `public bool MCI_Write(MCI_Protocol.StructName StName, ushort offset, byte payloadlength, byte[] payload, int timeoutMs, out byte errorCode)`

Name:	<code>bool MCI_Write(MCI_Protocol.StructName StName, ushort offset, byte payloadlength, byte[] payload, int timeoutMs, out byte errorCode)</code>
Purpose	Used to Write Structures/ Memory from device via NFC interface
Return :	True when Successful, False when Error.
Input parameters:	MCI_Protocol.StructName StName: Name of structure to be read or command to be executed. See here for more details Offset : Offset from a Structure or Memory Location PayloadLength: Length of payload to be read TimeoutMs : Timeout in MilliSeconds (default 4000ms)
Output:	Error code is stored in out byte errorCode. 0x00 for Success, non-zero value indicates failure.

4.6 `public string GetDllVersion()`

Name:	<code>string GetDllVersion()</code>
Purpose	Returns the current DLL version used.
Output :	Returns the current DLL version used.

4.7 `public bool ST25DV_ReadMultipleBlocks(byte startingBlockNumber, byte numberOfBlocks, out byte[] OutBuffer)`

Name:	<code>bool ST25DV_ReadMultipleBlocks(byte startingBlockNumber, byte numberOfBlocks, out byte[] OutBuffer)</code>
Purpose	Used to read EEPROM blocks on ST25 NFC device or dead battery information of iPERL, Dead battery information is stored in.
Return :	True when success, false when error
Output:	OutBuffer variable contains bytestream of the Read block.

5 Error Handling

Following Error Codes are used while using reading/writing operations over NFC.

```
{0x00 , "Success, No error" },
{0x01 , "MCI Error: Timeout" },
{0x02 , "MCI Error: Wrong CRC" },
{0x03 , "MCI Error: Wrong password" },
{0x04 , "MCI Error: Data not available" },
{0x05 , "MCI Error: Invalid access" },
{0x06 , "MCI Error: Command not supported" },
{0x07 , "MCI Error: Command not executed" },
{0x08 , "MCI Error: Unidentified" }
```

6 Structures and commands for iPEARL

Value *MCI_Protocol.StructName StName* used in Read / Write functions has following format:

```
public enum StructName
{
    Calibration,
    Configuration,
    Status,
    Power,
    LCD,
    unused1,
    unused2,
    Diagnostics,
    ProductionInfo,
    AckResponse,
    RadioParams,
    RadioInfo,
    Command
}
```