iID® service

Product:	microsensys iID®3000 PRO RFID interfaces
Product Code:	**-**
Revision:	**1.3**
Date:	2023-02-14

RESTful API Definition

iID® service provides RESTful APIs based on iID® reader library supporting microsensys RFID interfaces.

This documentation describes the different endpoints available on this service.

URL:	http://localhost:19813 (access using local IP address also possible)
Port:	19813
Protocol:	http / https
API-Key Name:	ApiKey
API-Key:	hL4bA4nB4yI0vI0fC8fH7eT6
Tested devices:	MICROSOFT® Windows 10/64bit, Zebra TC77, Samsung S8/S10
Supported host interface:	RS232 serial, USB, Bluetooth SPP
Version:	1.3
Release date:	2023-02-14
Dependencies	iIDReaderLibrary, iID android library

Content

Short history

This chapter includes a short history of modifications of iID® service RESTful APIs.

Date	Reason	Modification	Release Date / Version	FileName
2022-03	- first release		0.9	iID®service_v0.x.apk
2022-09	- public release	- document update	1.0	iID®service_v1.0.apk
2022-12	- update functionality	- incremented default timeout to 2 seconds - added path to update timeout value	1.1	iID®service_v1.1.apk
2023-02	- update functionality (backwards compatible)	- added option to set protocol type in connection settings	1.2	iID®service_v1.2.apk
2023-02	- release for both Desktop and Android	- changed character casing of JSON formats but structure and functions unchanged	1.3	iID®service_v1.3.apk (also in Play Store available) iIDservice.zip

Initialization

Connection Settings

This chapter includes endpoints to configure the interface parameters needed to communicate with the RFID reader.

/api/iidservice/interface/doc/CurrentSettings?&PortName={portName}***&PortType=***{portType}***&InterfaceType=***{interfaceType}***?&ProtocolType=***{protocolType}

Header :

Method: POST
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Request :

Parameter:

Type	Name	Description
String	portName	Port name or port path associated to the RFID reader
Integer	portType	Number representing the port type. See *PortType* section
Integer	interfaceType	Number representing the interface type. See *InterfaceType* section
Integer	protocolType	(Optional!) Number representing the protocol type. See *ProtocolType* section

Response :

success
status code: 200
response: JSON message with configured parameters. Format:
{
“portName”:”{portName}”,
“portType”:”{portType}”,
“interfaceType”:”{interfaceType}”
“protocolType”:”{protocolType}”
}
error
status code: 400
response: JSON message with status. Format:
{
“portName”:*parameterStatus*,
“portType”:*parameterStatus,*
“interfaceType”:*parameterStatus*
}
Being *parameterStatus*: “ok” / “error”

Description :

Configures the communication interface parameters to use. It identifies the interface used to communicate with the RFID reader.

/api/iidservice/interface/doc/CurrentSettings

Header :

Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Response :

success
status code: 200
response: JSON message with configured parameters. Format:
{
“portName”:”{portName}”,
“portType”:”{portType}”
“interfaceType”:”{interfaceType}”
“protocolType”:”{protocolType}”
}
error
status code: 404
response: empty JSON message

JSON-Values:

Type	Name	Description
String	portName	Port name or path of selected interface parameters
Integer	portType	Number representing the port type. See *PortType* section
Integer	interfaceType	Number representing the interface type. See *InterfaceType* section
Integer	protocolType	Number representing the protocol type. See *ProtocolType* section

Description :

Gets the current configured communication interface parameters being used to communicate with the RFID reader.

/api/iidservice/interface/doc/PossibleSettings

Header :

Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Response :

success
status code: 200
response: JSON message with configured parameters. Format:

```
{
    “portName”:[
        “{portName1}”, “{portName2}”, …
    ],
    “portType”:{
        “Serial”:0,
        “Bluetooth”:2,
        “USB”:4
    },
    “InterfaceType”:{
        “HF”:1356,
        “UHF”:868
    }
}
```

error
status code: 404
response: empty JSON message

Description :

Gets supported values for parameters to use with *CurrentSettings* POST API method. It also checks the available communication port available in local machine.
Note: This method does not check if the communication port corresponds to a RFID reader. It just returns all available communication ports.

Connection Functions

This chapter includes endpoints to configure the interface parameters needed to communicate with the RFID reader.

/api/iidservice/doc/Initialize

Header :
Method: POST
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key
Request :
Response :
success
status code: 200
response: JSON message *true* or *false* according to the result of the process
Description :
Initializes a DOC communication using the pre-configured interface parameters.

/api/iidservice/doc/Terminate

Header :
Method: POST
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key
Request :
Response :
success
status code: 200
response: JSON message *true* or *false* according to the result of the process
Description :
Terminates the communication with the RFID reader.

/api/iidservice/doc/IsConnected

Header :
- Method: GET
- Details:
  - Async Communication
  - 'Access-Control-Allow-Origin', '*'
  - 'Content-Type', 'application/x-www-form-urlencoded'
- Authentication: Use API Key Name and API Key

Request :

Response :
- success
  - status code: 200
  - response: JSON message *true* or *false* according to the result of the process

Description :

Returns the connection state. The service checks if a communication with the RFID reader is still possible.

Common Methods

RFID Reader Information

This chapter describes how to get the information of the RFID Reader used to read TELID® sensor dataloggers.

/api/iidservice/doc/ReaderInfo

Header :

Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Response :

success
status code: 200
response: JSON message RFID information. Format:
{
“ReaderID”:”{readerID}”,
“HwInfo”:”{hwInfo}”,
“FwInfo”:”{fwInfo}”,
“HexString”:”{hexString}“
}
error
status code: 404
response: Empty JSON message.

JSON-Values:

Type	Name	Description
Integer	readerID	Reader ID of connected RFID reader
String	hwInfo	Represents the Hardware version of connected RFID reader
String	fwInfo	Represents the Firmware version of connected RFID reader
String	hexString	Hex string representation of reader information
JSON	hwInfoDetails	Extra hardware information

Description :

Returns the RFID reader information connected to TELID®service (if communication established).

RFID functions

This chapter describes API points to scan for RFID transponders and access (if available) the internal memory.

/api/iidservice/doc/Identify

Header :

Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Response :

success
status code: 200
response: JSON message read parameters. Format:
{
"interfaceType":"{interfaceType}",
"scanResult":{scanResult}
}
error
status code: 404
response: Empty JSON message.

JSON-Values:

Type	Name	Description
Integer	interfaceType	Number representing the selected interface type. May be used to detect the structure of *scanResult*
JSON	scanResult	Represents the scan result. Depending on the selected interface type may have one of the structures described below

Description :

Searches for transponder (or transponders) near the RFID reader antenna and returns the information.

Possible formats for JSON *scanResult* message:

JSON-Format for HF (1356) :

```
{
    "interfaceType":”1356”,
    "scanResult”:{
        “TagID”:”{uidHex}”
    }
}
```

JSON-Values:

Type	Name	Description
String	uidHex	Hex representation of the UID of the found transponder

JSON-Format for UHF (868) :

```
{
    "interfaceType":”868”,
    "scanResult”:[{
        “TagID”:{
            “PC”:”{pc}”,
            “UII”:”{uii}”
        },
        “antennaNumber”:”{antNumber}”,
        “RSSI”:”{rssi}”
    },{…}
    ]
}
```

JSON-Values:

Type	Name	Description
String	PC	Hex representation of the PC bytes of the found transponder
String	UII	Hex representation of the UII bytes of the found transponder
Integer	antNumber	Number representing the antenna where the transponder was found
Float	Rssi	Number representing the RSSI value

/api/iidservice/doc/IdentifyCycle?&ApiKey={apiKey}***&IntervalMs=***{intervalMs}***&NumTries=***{numTries}

Details :

Server Sent Event
Connection type: keep-alive
Cache-Control: no-cache
Content-Type: text/even-stream
Reconnection time: 5s
Update time: *{intervalMs}* ms
Number of times to perform function: *{numTries}*

Events :

Event Name	Description
message	Raised to notify a new measurement read
completed	Raiesd when cycle process completed

Event details :

message
Parameters:
data: JSON message.
Format follows the *Identify* API above.

Description :

Establishes a connection to the service that cyclically perform *Identify* and reports back the found information.

/api/iidservice/doc/ReadBytes?&TagID={tagID}***&PageNum=***{pageNum}***&FromByte=***{fromByte}***&LengthBytes=***{lengthBytes}

Header :

Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Request :

Parameter:

Type	Name	Description
String	tagID	Hex representation of the transponder to read data from
Integer	pageNum	Memory page/bank number to be read (only supported by some transponder types)
Integer	fromByte	First memory position to be read (being 0 the first byte of the selected memory)
Integer	lengthBytes	Number of bytes to be read

Response :

success
status code: 200
response: JSON message with read data. Format:
{
“tagID”:”{tagID}”,
“pageNum”:”{pageNum}”,
“fromByte”:”{fromByte}”,
“lengthBytes”:”{lengthBytes}“,
“data”:”{dataHex}”
}
Parameters from request are return, and *dataHex* contains the hex representation of the read contents.
error-process
status code: 404
response: Empty JSON message.

Description :

Reads the transponder memory and returns the data.

/api/iidservice/doc/WriteBytes?&TagID={tagID}***&PageNum=***{pageNum}***&FromByte=***{fromByte}***&Data=***{data}

Header :

Method: POST
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Request :

Parameter:

Type	Name	Description
String	tagID	Hex representation of the transponder to write data to
Integer	pageNum	Memory page/bank number to be written (only supported by some transponder types)
Integer	fromByte	First memory position to be written (being 0 the first byte of the selected memory)
String	data	Hex representation of the data to be written into the transponder

Response :

success
status code: 200
response: JSON message with read data. Format:
{
“tagID”:”{tagID}”,
“pageNum”:”{pageNum}”,
“fromByte”:”{fromByte}”,
“data”:”{data}”
}
error-process
status code: 404
response: empty JSON message

Description :

Writes the given data in the desired transponder memory.

RFID sensor transponder functions

This chapter describes how to get sensor information from TELID® sensor transponders.

/api/iidservice/doc/telidtransponder/GetSensorData

Header :

Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Response :

success
status code: 200
response: JSON message with configured parameters. Format:

```
{
    “serialNumber”:”{serialNum}”,
    “serialNumberHex”:”{serialNum}”,
    “description”:”{description}”,
    “measurements”:[{
        “timestamp”:”{timestamp}”,
        “values”:[{
            “magnitude”:”{magnitude}”,
            “unit”:”{unit}”,
            “valueType”:”{valueType}”,
            “symbol”:”{symbol}”
        },{…}
        ]
    },{…}
    ]
}
```

error-process
status code: 404
response: Empty JSON message.

Description :

Reads sensor measurements from a TELID® sensor transponder.

JSON-Values:

Type	Name	Description
Integer	serialNum	Serial number of the TELID® transponder
String	serialNumHex	Hex representation of the serial number
String	description	Description of found TELID® transponder
String	timestamp	Timestamp of the measurement
Integer	magnitude	Sensor value magnitude
String	unit	Unit of sensor value
String	valueType	Represents the sensor value type
String	symbol	Symbol of sensor value

/api/iidservice/doc/telidtransponder/GetSensorDataCycle?&ApiKey={apiKey}***&IntervalMs=***{intervalMs}***&NumTries=***{numTries}

Details :

Server Sent Event
Connection type: keep-alive
Cache-Control: no-cache
Content-Type: text/even-stream
Reconnection time: 5s
Update time: *{intervalMs}* ms
Number of times to perform function: *{numTries}* (0 performs the process until the connection is cancelled)

Events :

Event Name	Description
message	Raised to notify a new measurement read
completed	Raised when process completed

Event details :

message
Parameters:
data: JSON message.
Format described previously under “JSON format”.

Description :

Establishes a connection to the service that cyclically perform *GetSensorData* and reports back the found information.

Extras

This chapter describes extra functions.

/api/iidservice/interface/doc/Timeout?&Value={newTimeout}

Header :

Method: POST
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Request :

Parameter:

Type	Name	Description
Integer	Value	New timeout value used on RFID functions in milliseconds

Response :

success
status code: 200
response: JSON message *true*
error
status code: 400
response empty JSON message

Description :

Changes the current internally used timeout. This timeout defines the maximum time the service tries to find a transponder and optionally perform the required operations.

/api/iidservice/interface/doc/Timeout

Header :

Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Response :

success
status code: 200
response: JSON message with currently active timeout in milliseconds

Description :

Gets the current configured timeout being used in RFID functions in milliseconds.

Script communication

Connection Settings

This chapter includes endpoints to configure the interface parameters needed to communicate with the RFID reader.

/api/iidservice/interface/spc/CurrentSettings?&PortName={portName}***&PortType=***{portType}

Header :

Method: POST
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Request :

Parameter:

Type	Name	Description
String	portName	Port name or port path associated to the RFID reader
Integer	portType	Number representing the port type. See *PortType* section

Response :

success
status code: 200
response: JSON message with configured parameters. Format:
{
“portName”:”{portName}”,
“portType”:”{portType}”,
}
error
status code: 400
response: JSON message with status. Format:
{
“portName”:*parameterStatus*,
“portType”:*parameterStatus,*
}
Being *parameterStatus*: “ok” / “error”

Description :

Configures the communication interface parameters to use. It identifies the interface used to communicate with the RFID reader.

/api/iidservice/interface/spc/CurrentSettings

Header :

Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Response :

success
status code: 200
response: JSON message with configured parameters. Format:
{
“portName”:”{portName}”,
“portType”:”{portType}”
}
error
status code: 404
response: empty JSON message

JSON-Values:

Type	Name	Description
String	portName	Port name or path of selected interface parameters
Integer	portType	Number representing the port type. See *PortType* section

Description :

Gets the current configured communication interface parameters being used to communicate with the RFID reader.

/api/iidservice/interface/spc/PossibleSettings

Header :

Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Response :

success
status code: 200
response: JSON message with configured parameters. Format:

```
{
    “portName”:[
        “{portName1}”, “{portName2}”, …
    ],
    “portType”:{
        “Serial”:0,
        “Bluetooth”:2,
        “USB”:4
    },
}
```

error
status code: 404
response: empty JSON message

Description :

Gets supported values for parameters to use with *CurrentSettings* POST API method. It also checks the available communication port available in local machine.
Note: This method does not check if the communication port corresponds to a RFID reader. It just returns all available communication ports.

Connection Functions

This chapter includes endpoints to configure the interface parameters needed to communicate with the RFID reader.

/api/iidservice/spc/Initialize?&SpcDataPrefix={spcPrefix}***&SpcDataSufix=***{spcSufix}***&UseMssProtocol=***{mssProtocol}

Header :
Method: POST
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key
Request :
Response :
success
status code: 200
response: JSON message *true* or *false* according to the result of the process
Description :
Initializes a SPC communication using the pre-configured interface parameters and the given SPC communication parameters (which will be used to decode the data sent by the script running on the RFID reader).

/api/iidservice/spc/Terminate

Header :
Method: POST
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key
Request :
Response :
success
status code: 200
response: JSON message *true* or *false* according to the result of the process
Description :
Terminates the communication with the RFID reader.

/api/iidservice/spc/IsInitialized

Header :
Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key
Request :
Response :
success
status code: 200
response: JSON message *true* or *false* according to the result of the process
Description :
Returns the initialization state of the SPC communication.

/api/iidservice/LastHeartbeat

Header :
Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key
Request :
Response :
success
status code: 200
response: JSON message with configured parameters. Format:
{
“readerID”:”{readerID}”,
“scriptName”:”{scriptName}”,
“batteryStatus”:”{batteryStatus}”,
“timestamp”:”{ timestamp }
}
error-process
status code: 404
response: Empty JSON message.

JSON-Values:

Type	Name	Description
Integer	readerID	ID of the RFID reader
String	batteryStatus	Text representing the current battery status
String	timestamp	Timestamp of the last heartbeat update

Description :
Obtains the last heartbeat information sent by the script running on the RFID reader.

SPC send and receive methods

This chapter describes how to get the information of the service itself and its underlying library.

/api/telidservice/spc/ReceiveUpdates?&ApiKey={apiKey}***&HeartbeatTimeoutMs=***{timeoutMs}***&UpdateReportIntervalMs=***{updateIntervalMs}

Details :

Server Sent Event
Connection type: keep-alive
Cache-Control: no-cache
Content-Type: text/even-stream
Reconnection time: 5s
Heartbeat timeout: {timeoutMs} (default = 5s)
Update time: {updateIntervalMs} (default 500ms)

Events :

Event Name	Description
open	Raised when EventSource connection is established
error	Raised when an error occurs
message	<Not used>
readerHeartbeat	Raised to notify a new heartbeat received
readerRawData	Raised to notify a new data package received

Event details :

readerHeartbeat
Parameters:
id: Number of reported data packages. Is automatically increased each time the event is raised.
-1 → Error connecting to reader or connection lost.
data: JSON message reader heartbeat.
Format described below under “JSON format”.

readerRawData
Parameters:
id: Number of reported data packages. Is automatically increased each time the event is raised
data: JSON message reader raw data.
Format described below under “JSON format”.

Description :

Establishes a connection to the service that notifies data received from the script.

JSON Format :

reader heartbeat message

```
{
    “readerID”:”{readerID}”,
    “scriptName”:”{scriptName}”,
    “battery”:”{batteryStatus}”,
    “timestamp”:”{timestamp}
}
```

Raw data message

```
{
    “readerID”:”{readerID}”,
    “data”:”{data}”,
    “timestamp”:”{timestamp}”
}
```

JSON-Values:

Type	Name	Description
Integer	readerID	ID of the RFID reader
String	batteryStatus	Text representing the current battery status
String	lastUpdate	Timestamp of the last heartbeat update
String	Timestamp	Timestmap when the data was received
String	Data	Data received from script

/api/iidservice/spc/Receive

Header :

Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Request :

Response :

success
status code: 200
response: JSON message with configured parameters. Format:
{
"ReaderID":"{readerID}",
"Data":"{data}",
"Timestamp":"{timestamp}"
}
no data available
status code: 404
response: empty JSON message

Description :

Obtains last received SPC data from the script running on the RFID reader.

/api/iidservice/spc/SendSpcRequestASCII?&Request={request}

Header :

Method: POST
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Request :

Parameter:

Type	Name	Description
String	request	Request content to send to the running script

Response :

success
status code: 200
response: empty JSON message
error-parameters
status code: 400
response: empty JSON message

Description :

Sends the data as SPC request to the script running on the RFID reader.

/api/iidservice/spc/SendSpcRequestBytes?&RequestHex={requestHex}

Header :

Method: POST
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Request :

Parameter:

Type	Name	Description
String	requestHex	Request content in HEX format to send to the running script

Response :

success
status code: 200
response: empty JSON message
error-parameters
status code: 400
response: empty JSON message

Description :

Sends the data as SPC request to the script running on the RFID reader.

Library functions

iID® reader library methods

This chapter describes how to get the information of the service itself and its underlying library.

/api/iidservice/Version

Header :
Method: GET
Details:
Async Communication
'Access-Control-Allow-Origin', '*'
'Content-Type', 'application/x-www-form-urlencoded'
Authentication: Use API Key Name and API Key

Response :
success
status code: 200
response: JSON message with configured parameters. Format:
{
“libraryVersion”:”{libVersion}”,
“serviceVersion”:”{serviceVersion}”,
“serviceReleaseDate”:”{serviceReleaseDate}”
}

Description :
Returns the iID® reader service version information.