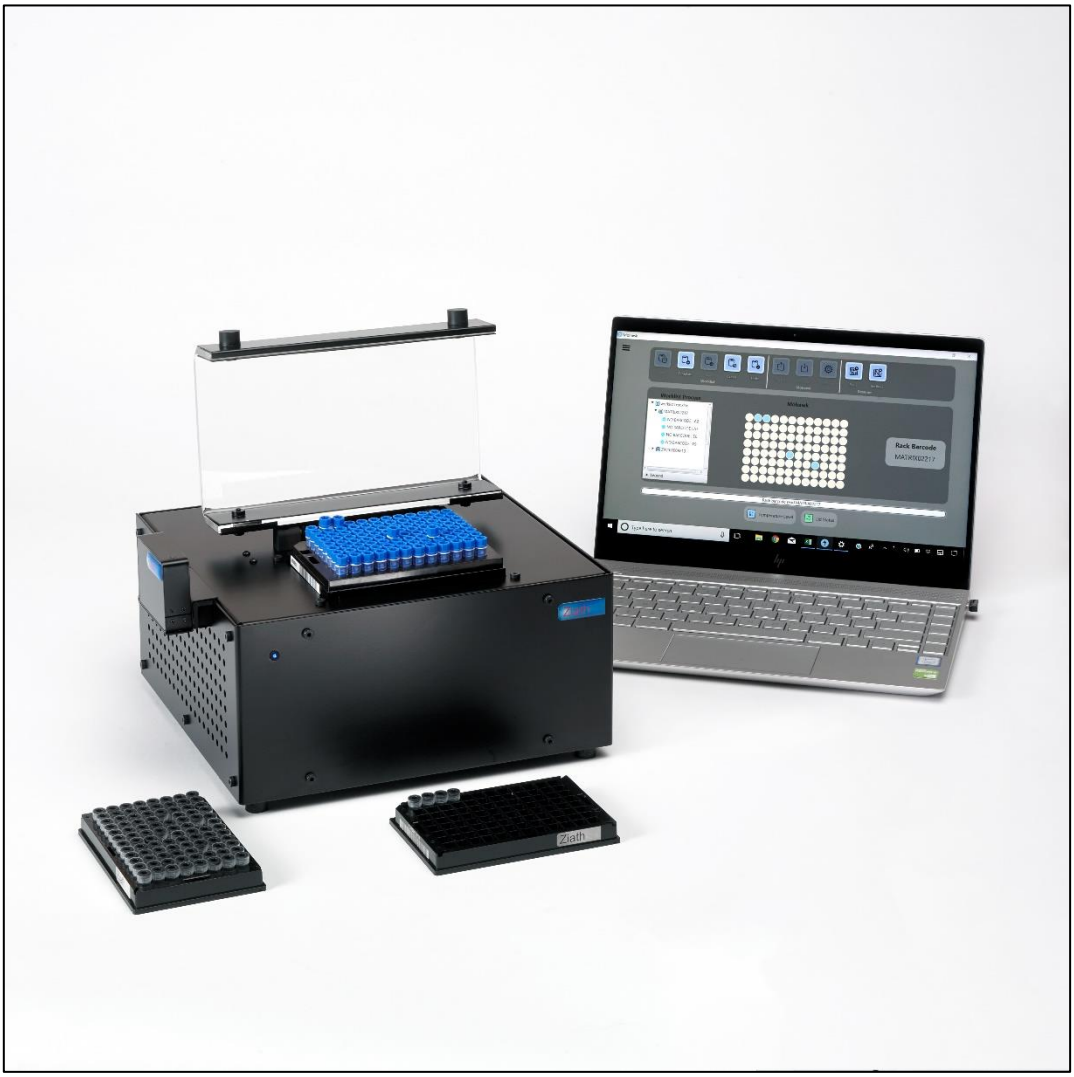
Mohawk Remote Control Guide



Version 2.5

Mohawk™ Remote Control Guide

This is Version 2.5 of the Mohawk™ Remote Control Guide

Contents

Mohawk Remote Control

Introduction

Mohawk enables programmatic access to Mohawk through two different APIs, each of them allow the interaction of a separate computer program with the Mohawk and control the device without a Graphical User Interface showing on the screen.

The new Mohawk Remote API provides a REST API for Mohawk control and a WebSocket endpoint for Mohawk notifications.

The old TCP/IP Mohawk Remote Control provides interfaces to raw TCP sockets for Mohawk control and notifications.

Mohawk Remote API

Mohawk Remote API is enabled by a separate program entitled ***WebServer.exe*** which is provided. This starts the REST API that uses HTTP requests to access and control Mohawk Software and the Mohawk WebSocket endpoint which creates a persistent connection between the server and the client where the notifications are published.

Java Sample code

Ziath provides sample code in Java for integration use.

https://github.com/ZiathLtd/MohawkWebserverJava

Python Sample Code

Ziath provides sample code in Python for integration use.

https://github.com/ZiathLtd/MohawkWebserverPython

.NET Sample Code

Ziath provides sample code in .NET for integration use.

https://github.com/ZiathLtd/MohawkServerDotnetExample

Start Mohawk in Remote Mode

In the Program Files directory of Mohawk there is an application call MohawkWebServer.exe. Once this is started the system is ready to go and receive RESTful commands. Note that the Mohawk GUI and webserver cannot be running at the same time.

API base URLs

All URLs for RESTful operations are served on port 8556 for HTTP connections. Therefore if controlled from the same computer all calls will be prepended by http://localhost:8556. In addition notifications can be sent via a websocket communication system, again if on the same computer this is available on port XXXX so websocket requests are received on ws://localhost.

Model

The elements below defined the various JSON components which can be sent and/or received on a RESTful call or websocket event.

Pin

```
{
        “row” : number,
        “column” : number,
        “pin_up”: boolean

}
```

ReaderConf

```
{
“type”: string
}
```

RackLoader

```
{
“rack_barcode”: string,
“reset_pins”: boolean
}
```

Worklist

```
{
“items” : List of PickItem,
“path” : path,
“load_time” : string,
“finished_time”: string,
“finished” : Boolean
}
```

PickItem

```
{
        “rack_barcode”: string,
        “tube_barcode”: string,
        “row”: number,
        “column”: number,
        “picked”: boolean
}
```

ErrorInfo

```
{
        “message”: string,
        “error”: string,
        “type”: string,
```

“timestamp”: string
}

ScanList

{
“timestamp”: string
“scanItems”: {ScanItem}
}

ScanItem

{
“tubeBarcode”: string,
“row”: number,
“column”: number,
“success”: boolean,
“inPicklist”: boolean,
}

Endpoints

There are a series of endpoints, each of which relate to a discrete function of the mohawk software and/or hardware.

Version

This will return the running Mohawk software version.

Method*:* GET
URL*:* /mohawk/api/v1/version
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json
Body: {“result”: “<VERSION_STRING>”}
Example: {“result”: “2.4”}

Status Code: 417
Body: **ErrorInfo** JSON object

Format

This will return the format of the mohawk, this defines if the device is configured for 96 or 48 well racks

Method*:* GET
URL*:* /mohawk/api/v1/format
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json
Body: {“result”: <48/96>}

Status Code: 417
Body: **ErrorInfo** JSON object

Mohawk Temperature

This returns the internal temperate of the Mohawk in Celsius.

Method*:* GET
URL**:** /mohawk/api/v1/temperature
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json
Body: {“result”: <TOP_TEMPERATURE_DEGREES>}

Status Code: 417
Body: **ErrorInfo** JSON object

Mohawk Fan Speed

This returns the current speed of the fan as a number between 0 and 255 with 0 being off and 255 max speed.

Method**:** GET
URL*:* /mohawk/api/v1/fan_speed
Parameters**:** None

Responses

Status Code: 200
Media Type: application/json
Body: {“result”: <FAN_SPEED>}

Status Code: 417
Body: **ErrorInfo** JSON object

Mohawk Lid Status

This returns if the lid of the scanner is open or not, OPEN for open or CLOSED for closed.

Method*:* GET
URL*:* /mohawk/api/v1/lid_status
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json
Body: {“result”: “<OPEN|CLOSED>”}

Status Code: 417
Body: **ErrorInfo** JSON object

Mohawk Status

This will return the current status of the mohawk activity. The values are as follows:

- IDLE – the unit is waiting for picklist
- PICKING – the unit has a picklist ready but is not actively in operation
- BUSY – the unit is actively in operation
- ERROR – the unit cannot operate due to a system error

Method*:* GET
URL*:* /mohawk/api/v1/mohawk_status
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json
Body: {“result”: “<IDLE | BUSY | PICKING | ERROR>”}

Status Code: 417
Body: **ErrorInfo** JSON object

Pin Status

This will return a list of all pins in the system and if they are currently up or down.

Method*:* GET
URL*:* /mohawk/api/v1/pins_status
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json
Body: Array of **Pin** objects
Body Example: [
{

```
    "row": 1,
    "column": 1,
    "pin_up": false
  },
  {
    "row": 1,
    "column": 2,
    " pin_up ": false
  },
  …
  {
    "row": 8,
    "column": 12,
    " pin_up ": false
  }
]
```


Status Code: 417
Body: **ErrorInfo** JSON object

Pins Up

This will instruct the mohawk to fire some pins up. Note that only 16 pins can be 'up' at any one time; if this is exceeded an ErrorInfo object is returned.

Method*:* POST
URL*:* /mohawk/api/v1/pins_up
Parameters*:*

- Request Body:
  - Array of **Pin** objects

Responses

Status Code: 200
Media Type: application/json
Body: Array of **Pin** objects
Body example: [


```
  {
    "row": 1,
    "column": 2
  },
 …
  {
    "row": 2,
    "column": 3
  }
]
```

Status Code: 417
Body: **ErrorInfo** JSON object

Reset Pins

This will force all pins to drop immediately, if all the pins are down the machine will not do anything and return OK.

Method*:* POST
URL*:* /mohawk/api/v1/reset_pins
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json
Body: {“result”: “OK”}

Status Code: 417
Body: **ErrorInfo** JSON object

Configure linear reader

This will set the linear reader to either be a manual or ziath linear reader and will return the current linear reader type.

Method*:* POST
URL*:* /mohawk/api/v1/reader
Parameters*:*

- Request Body:
  - **ReaderConf** object
  - Example: {“type”: “ZIATH” } or {“type”: “MANUAL” }

Responses

Status Code: 200
Media Type: application/json
Body: {“type”: “ZIATH” } or {“type”: “MANUAL” }

Status Code: 417
Body: **ErrorInfo** JSON object

Read Rack Barcode

This will return the currently read barcode on the Mohawk by the linear scanner. IF there is no linear scanner plugged in it returns an ErrorInfo object.

Method*:* POST
URL*:* /mohawk/api/v1/read_barcode
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json
Body: {“result”: “<RACK_BARCODE>”|”NO_READ”}

Status Code: 417
Body: **ErrorInfo** JSON object

Set Rack barcode

Sets the rack barcode in the system if the linear reader is not installed, if there is a linear reader installed an ErrorInfo object will be returned. If reset pins is true all pins will drop at the same time as this is set.

Method*:* POST
URL*:* /mohawk/api/v1/set_rack_barcode
Parameters*:*

- Request Body:
  - **RackLoader** object
  - Example: {“rack_barcode”: “001”, “reset_pins”: true}

Responses

Status Code: 200
Media Type: application/json
Body: {“result”: “OK”}

Status Code: 417
Body: **ErrorInfo** JSON object

Load Worklist

This will load a prewritten worklist into the Mohawk control software. The format of the worklist is as in the Mohawk GUI. Note that there are different endpoints for the different formats in which a worklist can be loaded. In addition, there is an extra JSON format which is only available in the RESTful API. If there is already a workist loaded the previous worklist is discarded.

This JSON format is defined as XXXXX

Method: POST
URL*:* /mohawk/api/v1/worklist/load_<xml|excel|csv|json>
/mohawk/api/v1/worklist/load_xml
/mohawk/api/v1/worklist/load_csv
/mohawk/api/v1/worklist/load_excel

/mohawk/api/v1/worklist/load_json

Parameters*:*

- Request Body:
  - **Binary**

Responses

Status Code: 200
Media Type: application/json
Body: {"result": "OK"}

Status Code: 417
Body: **ErrorInfo** JSON object

Worklist Status

This will return the current status of a loaded worklist which are either:

- NO_WORKLIST – no worklist is loaded or the previous worklist has been cancelled
- LOADED – a worklist is loaded and being processed
- FINISHED – a worklist has been loaded and finished

Method*:* GET
URL*:* /mohawk/api/v1/worklist/status
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json
Body: {"result": NO_WORKLIST | LOADED | FINISHED}

Worklist Progress

This will return the current progress of the worklist, if no worklist is loaded an ErrorInfo object will be returned.

Method*:* GET
URL*:* /mohawk/api/v1/worklist
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json
Body: **Worklist** object

Status Code: 417
Body: **ErrorInfo** JSON object

Finish Worklist

This will instruct the system to complete the currently loaded worklist, if there is no worklist loaded an ErrorInfo object will be returned.

Method*:* POST
URL*:* /mohawk/api/v1/worklist/finish
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json
Body: {“result”: “OK”}

Status Code: 417
Body: **ErrorInfo** JSON object

Cancel Worklist

This will instruct the system to discard the currently loaded worklist; no reports can be generated from this worklist after it has been cancelled. If no worklist is loaded an ErrorInfo object will be returned

Method*:* POST
URL*:* /mohawk/api/v1/worklist/cancel
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json
Body: {“result”: “OK”}

Status Code: 417
Body: **ErrorInfo** JSON object

Generate Report

This will generate a report for the last executed worklist, if there is no worklist loaded an ErrorInfo object is returned.

Method*:* GET
URL*:* /mohawk/api/v1/report_to_<xml|excel|csv|json>
Parameters*:* None

Responses

Status Code: 200
Media Type: application/json or application/xml or application/text

Shutdown

This will shutdown the mohawk once any physical operations with the device have completed.

Method: POST
URL: /mohawk/api/v1/shutdown
Parameters: None

Responses

Status Code: 200
Media Type: application/json
Body: {“result”: “OK”}

Force Shutdown

This will immediate shutdown the mohawk software regardless of what the Mohawk device is doing.

Method: POST
URL: /mohawk/api/v1/force_shutdown
Parameters: None

Responses: None

Websockets

There are a number of instances when controlling the Mohawk which require a notificatuion without a prompt from the calling client. In this case a simepl RESTful call will not suffice. The de facto method for enabling such a system is Websockets (https://en.wikipedia.org/wiki/WebSocket). Mohawk has a websocket system that can be used to receive notification of events from the device. Websockets can be implemented in multiple languages, there are Java, Python and C# examples in the public git repositories detailed earlier in this document.

When an event is triggered by the Mohawk a Notification object will be encoded. This object contains a payload with annotation on the event and a NotificiationMessage which has many options that are detailed below:

rackNotInWorklist

Description - A rack has been placed on the Mohawk which is not in the loaded worklist.
Payload – the barcode of the rack in question as a String

rackPicked

Description - A rack has been placed on the Mohawk and tubes in the rack have been picked
Payload – A set of the tubes as PickItem Models

rackBarcodeNotReadable

Description – a rack has been placed on the Mohawk but no barcode can be read
Payload - None

rackAlreadyPicked

Description – a rack has been placed on the Mohawk to be picked but it has already been picked
Payload - the barcode of the rack in question as a String

newBarcodeRead

Description – a rack has been placed on the Mohawk and the barcode has been read
Payload - the barcode of the rack in question as a String

linearReaderConnected

Description – the linear reader has been connected to the Mohawk
Payload - None

linearReaderDisconnected

Description – the linear reader has been disconnected from the Mohawk
Payload - None

lidUp

Description – the lid of the Mohawk has been raised
Payload - None

lidDown

Description – the lid of the Mohawk has been closed
Payload - None

tempOverMaximum

Description – the internal temperature of the Mohawk has gone above the allowed maximum, the Mohawk will shutdown until the temperature reduces
Payload - None

tempBelowMaximum

Description – the internal temperature of the Mohawk has gone above the allowed maximum and subsequently lowered again
Payload - None

worklistCompleted

Description – a worklist has been loaded and subsequently completed
Payload - None

pinsReset

Description – the pins of the mohawk have been reset (dropped down)
Payload - None

firmwarePinReset

Description – the pins of the mohawk have been reset (dropped down) due to them having been up for too long without a command to reset. This time is 60 seconds.
Payload - None

mohawkConnected

Description – the mohawk RS232 communication has been connected
Payload - None

mohawkDisconnected

Description – the mohawk RS232 communication has been disconnected
Payload - None

linearReaderNotReady

Description – the linear ready has been connected but is not ready to read a barcode
Payload - None

linearReaderUnplugged

Description – the linear reader has been unplugged after having been plugged in and connected
Payload – None

linearReaderPlugIn

Description – the linear reader has been plugged in
Payload – None

scanResult

Description – DP5 or Datapaq has scanned a rack and the results are available
Payload – the rack barcode and a ScanList

scanException

Description – DP5 or Datapaq has scanned a rack and there is an error
Payload – a string describing the error

TCP/IP Mohawk Remote Control

TCP/IP Mohawk remote control is enabled by a separate program entitled *MohawkServer.exe* which is provided. This starts a command server and a notification server.

.NET Library

Ziath provides a .NET library for integration use; contact support@ziath.com for a reference.

Jar Library

Ziath provides a Java jar for integration use; contact support@ziath.com for a reference.

Python Sample Code

Ziath provides sample code in Python for integration use; contact support@ziath.com for a reference.

Initialize Mohawk Remote Control

The server mode starts the Mohawk control software, listens on a specified port for incoming commands and publish notifications in another port. Multiple clients can attach to the Mohawk control software; however, only one operation at a time can be performed. The Mohawk control software will start and listen to a specified port for commands (by default the server will listen on port 8555) and will allow connections for notifications in another port (by default 8282). However, these ports can be changed with the following command options:

Short Form	Long Form	Params	Default	Description
p	command-port	port number	8555	The port that the server will listen on.
n	notification-port	port number	8282	The port that the notification server will transmit messages on

Once the server is running in socket mode, the process will continue to listen to the socket until it is either closed or receives the command <shutdown/>. Note that there is a maximum number of simultaneous client connections; the amount can be discovered by calling <getMaxConnections/>. If this number is exceeded a string describing the current connections is returned and the socket closed from the server side. An example of a command line to start the server is below:

```
MohawkServer -p 8899
```

This will start the server running and it will listen for incoming connections on port 8899.

The server responds to a set of commands: all commands must be terminated by a carriage return and a line feed; all responses will be terminated with a carriage return and a line feed.

Should you need to close the server down and for some reason the server is not responding then if you connect to port 9933 then the server will immediately close down and close the connection. This is designed to be an emergency shutdown and should not typically be used to close the server.

Connection

When the Mohawk control software is first connected to the initial response of <info>Mohawk Remote Server 1.0</info> will be returned. The 1.0 is the version of the control software and will change as new software versions are released.

Commands

The typical command is an XML snippet indicating an instruction or querying a piece of information. A simple command will return with OK in a result XML element and queried information will also be returned in a result XML element snippet. If there is an error then the XML element of error will be returned. General errors such as 'Server Busy' and ' Command snippet not valid xml' apply to most commands and some commands have specific errors which are detailed on the command description. Note that all commands are expressed as xsd in Appendix A.

Version

Command : <version/>
Reply : <result>1.0</result>
Errors : None
Since : Version 1.0

This will return the running Mohawk software version.

Fan Speed

Command : <fanSpeed/>
Reply : <result>the fan speed between 0 and 255</result>
Errors : 8 (device communication error)
Since : Version 1.0

This will return the speed of the fan inside the Mohawk; 0 is off and 255 is running at full speed. Currently the Mohawk only runs at the speed of off or full speed but this may change in the future.

Temperature

Command : <temperature/>
Reply : <result>the temperature of the solenoid array</result>
Errors : 8 (device communication error)
Since : Version 1.0

This will return the temperature of the solenoid array in the Mohawk.

Load Worklist

Command : <loadWorklist path="location of worklist file" type="xml|excel|json|csv"/>
Reply : <result>OK</result>
Errors : 7 (path not found), 12 (worklist already loaded), 9 (worklist validation error), 11 (worklist load error)
Since : Version 1.0

This will load the specified worklist.

Load Rack

Command : <loadRack barcode=”rack-barcode” reset=”true|false”/>
Reply : <result>OK</result>
Errors: 13 (no worklist loaded)
Since : Version 1.0

This will load a rack in the software and if the rack is in the worklist the pins for the rack will fire. Note that a worklist must be loaded for this to work. If reset is passed then the rack will not be marked as already picked.

Worklist Status

Command : <worklistStatus/>
Reply : <result>LOADED</result>
Errors : None
Since : Version 1.0

This will send back the current status of a running worklist. The reply would be one of:

- NO_WORKLIST
- FINISHED
- LOADED

Finish Worklist

Command : </finishWorklist>
Reply : <result>OK</result>
Errors : 13 (worklist not loaded)
Since : Version 1.0

This will mark a worklist as finished in the system; note that if you do not have a worklist loaded then the above error will be returned.

Generate Report

Command : <generateReport path=”the path and filename of the report” type=”xml|excel|json|csv”/> (note that the path is optional and if omitted the report will be sent to the client via the socket connection)

Reply : if a path is supplied <result>OK</result> otherwise <result>the report</result> note that if the report is requested to be Excel the entire excel report will be returned as bytes

Errors : 7 (file path not found), 15 (worklist not finished), 10 report generation error)

This will generate a report if a worklist is finished; note that if a path is provided the report will be written to the file. If you request CSV then two files will be written to a folder and XML, JSON and Excel will write to one file. If you do not enter a path then the report will be passed to the client along the socket connection.

Lid Status

Command : <lidStatus/>
Reply : <result>OPEN|CLOSED</result>
Errors : 8 (device communication error)

Since : Version 1.0

This will return the status of the lid; open or closed

Mohawk Status

Command : <mohawkStatus/>
Reply : <result>IDLE|BUSY|PICKING|ERROR</result>
Errors : None
Since : Version 1.0

This will return the status of the Mohawk. Note that this is one command which will always execute, even if the Mohawk is 'busy'.

Linear Reader

Command : <linearReader type="ZIATH|MANUAL"/>
Reply : <result>OK</result>
Errors : None
Since : Version 1.0

Command : <linearReader/>
Reply : <result>ZIATH</result>
Errors : None
Since : Version 1.0

This will set the linear reader to either be a manual or ziath linear reader. Alternatively if the type is not entered then the currently configured linear scanner will be returned.

Current Connections

Command: <currentConnections/>
Reply : <result>3</result>
Errors: None
Since: Version 1.0

This will return the number of currently connected clients to the server

Read Rack Barcode

Command : <readRackBarcode/>
Reply : <result>rackbarcode|NO_READ</result>
Errors : 16 (Read rack barcode error)
Since : Version 1.0

This will read the rack barcode on the Mohawk. If no barcode can be read NO_READ will be returned.

Max Connections

Command: <maxConnections/>
Reply : <result>20</result>
Errors: None
Since: Version 1.0

This will return the maximum number of connected clients to the server

This will load the worklist in; note that the file must be syntactically correct, present and a worklist not already running.

Worklist Progress

Command: <worklistProgress/>
Reply: <result>a full report in xml format</result>
Errors: 10 (report generation error), 13 (worklist not loaded)
Since: Version 1.0

If this is called a full report in xml format will be returned

Pin Status

Command : <pinStatus/>
Reply : <result>{<pinStatus row=”pinrow” column=”pinColumn” pinup=”true|false”/>}</result>
Errors : 8 (driver communication error)
Since : Version 1.0

This will return the status of all of the pins on the Mohawk as discrete XML elements for each pin.

Force Pins

Command : <forcePins><pin row=”1” column=”2”/></forcePins>

Reply : <result>OK</result>
Errors : 3 (driver error), 4 (lid open), 5 (force pins failed)
Since : Version 1.0

If this command is entered then all pins entered will be fired.

Reset Pins

Command : <resetPins/>
Reply : <result>OK</result>
Errors : 8 (device communication error)
Since : Version 1.0

This will drop all pins on the Mohawk immediately.

Close

Command : <close/>
Reply : <result>OK</result>
Errors : None
Since : Version 1.0

This will close your client’s connection to the server; note that it will keep the server running but just close your client connection. Before it is closed the return will be sent back to indicate a successful command.

Force Close

Command : <forceClose/>
Reply : <result>OK</result>
Errors : None
Since : Version 1.0

This will close your client's connection to the server; note that it will keep the server running but just close your client connection. Before it is closed the return will be sent back to indicate a successful command. Note that this will ignore any running status on the Mohawk and close your connection.

Shutdown

Command : <shutdown/>
Reply : <result>OK</result>
Errors : None
Since : Version 1.0

This will shut the system down.

Force Shutdown

Command : <forceShutdown/>
Reply: None
Since : Version 1.0

This will immediately terminate the mohawk control software; no acknowledgement will be given and no regard given for the state of the Mohawk

Notification Server

This interface will inform the client of events that have happened on the Mohawk which were not triggered by the action of the client but by the user of the Mohawk. This works over a different port to the command port and will not reply to or aknowledge any commands; from the point of view of the client it is read only.

Firmware Pins Reset

Event: <firmwarePinReset/>

This is sent when the pin is left up and the Mohawk has dropped the pins after a preset time of 2 minutes.

Lid Down

Event: <lidDown/>

This is sent when the user drops the lid on the Mohawk

Lid Up

Event: <lidUp/>

This is sent when the user raises the lid on the Mohawk

New Barcode Read

Event: <newBarcodeRead><![CDATA[ZI878765]]></ newBarcodeRead>

This is sent when the linear reader detects a new barcode; note that if a rack is removed then NO_READ will be sent as a barcode.

Rack Already Picked

Command: <rackAlreadyPicked/>

This is sent when the user has placed a rack on the Moahwk and dropped the lid but the rack has already been picked.

Rack Barcode Not Readable On Lid Down

Event: <rackBarcodeNotReadable/>

This is triggered when the lid is closed and no readable barcode is present

Rack Not in Worklist

Event: </rackNotInWorklist><![CDATA[ZI878765]]></rackNotInWorklist>

This is triggered when the rack on the picker is not in the currently loaded worklist.

Rack Picked

Event: <rackPicked><pin row=”1” column=”2”/><pin row=”3” column=”4/></pin>

This is triggered when the pins of a rack are fired having been in a worklist.

Temp Over Maximum

Event: <tempOverMaximum/>

This is triggered when the picker has over heated due to excess use.

Temp Under Maximum

Event: <tempUnderMaximum/>

This is triggered after an over temp event and is triggered when the temperature drops back down again

Appendix A

XSD Formats

Each command will be expressed in XSD format below:

Version

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="getVersion"></xs:element>
</xs:schema>
```

Fan Speed

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="fanSpeed"></xs:element>
```

```
</xs:schema>
```

Temperature

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">

        <xs:element name="temperature">
                <xs:complexType>
                        <xs:attribute name="level" type="levelType"></xs:attribute>
                </xs:complexType>
        </xs:element>

        <xs:simpleType name="levelType">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="top" />
                        <xs:enumeration value="bottom" />
                </xs:restriction>
        </xs:simpleType>

</xs:schema>
```

Load Worklist

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="loadWorklist">
                <xs:complexType>
                        <xs:attribute name="path" type="xs:string" use="required" />
                        <xs:attribute name="type" type="worklistType" use="required" />
                </xs:complexType>
        </xs:element>

        <xs:simpleType name="worklistType">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="xml" />
                        <xs:enumeration value="excel" />
                        <xs:enumeration value="json" />
                        <xs:enumeration value="csv" />
                </xs:restriction>
        </xs:simpleType>
</xs:schema>
```

Load Rack

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="loadRack">
                <xs:complexType>
                        <xs:attribute name="barcode" type="xs:string" use="required" />
                        <xs:attribute name="reset" type="xs:boolean" use="required" />
                </xs:complexType>
        </xs:element>
</xs:schema>
```

Worklist Status

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="worklistStatus"></xs:element>
</xs:schema>
```

Finish Worklist

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="finishWorklist"></xs:element>
</xs:schema>
```

Generate Report

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="generateReport">
                <xs:complexType>
                        <xs:attribute name="path" type="xs:string" />
                        <xs:attribute name="type" type="worklistType" use="required" />
                </xs:complexType>
        </xs:element>

        <xs:simpleType name="worklistType">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="xml" />
                        <xs:enumeration value="excel" />
                        <xs:enumeration value="json" />
                        <xs:enumeration value="csv" />
                </xs:restriction>
        </xs:simpleType>
</xs:schema>
```

Lid Status

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="lidStatus"></xs:element>
</xs:schema>
```

Mohawk Status

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="mohawkStatus"></xs:element>
</xs:schema>
```

Linear Reader

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="linearReader">
                <xs:complexType>
                        <xs:attribute name="type" type="linearReaderType"/>
                </xs:complexType>
        </xs:element>

        <xs:simpleType name="linearReaderType">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="ziath" />
                        <xs:enumeration value="manual" />
                </xs:restriction>
        </xs:simpleType>
</xs:schema>
```

Read Rack Barcode

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="readRackBarcode"></xs:element>
</xs:schema>
```

Current Connections

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="currentConnections"></xs:element>
</xs:schema>
```

Max Connections

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="maxConnections"></xs:element>
</xs:schema>
```

Pin Status

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="pinStatus"></xs:element>
</xs:schema>
```

Force Pins

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="forcePins">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="pin" maxOccurs="unbounded">
                                        <xs:complexType>
                                                <xs:attribute name="row" type="xs:int" use="required"
/>
                                                <xs:attribute       name="column"       type="xs:int"
use="required" />
                                        </xs:complexType>
                                </xs:element>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
</xs:schema>
```

Reset Pins

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="resetPins"></xs:element>
</xs:schema>
```

Close

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="close"></xs:element>
</xs:schema>
```

Force Close

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="forceClose"></xs:element>
```

```
</xs:schema>
```

Shutdown

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="shutdown"></xs:element>
</xs:schema>
```

Force Shutdown

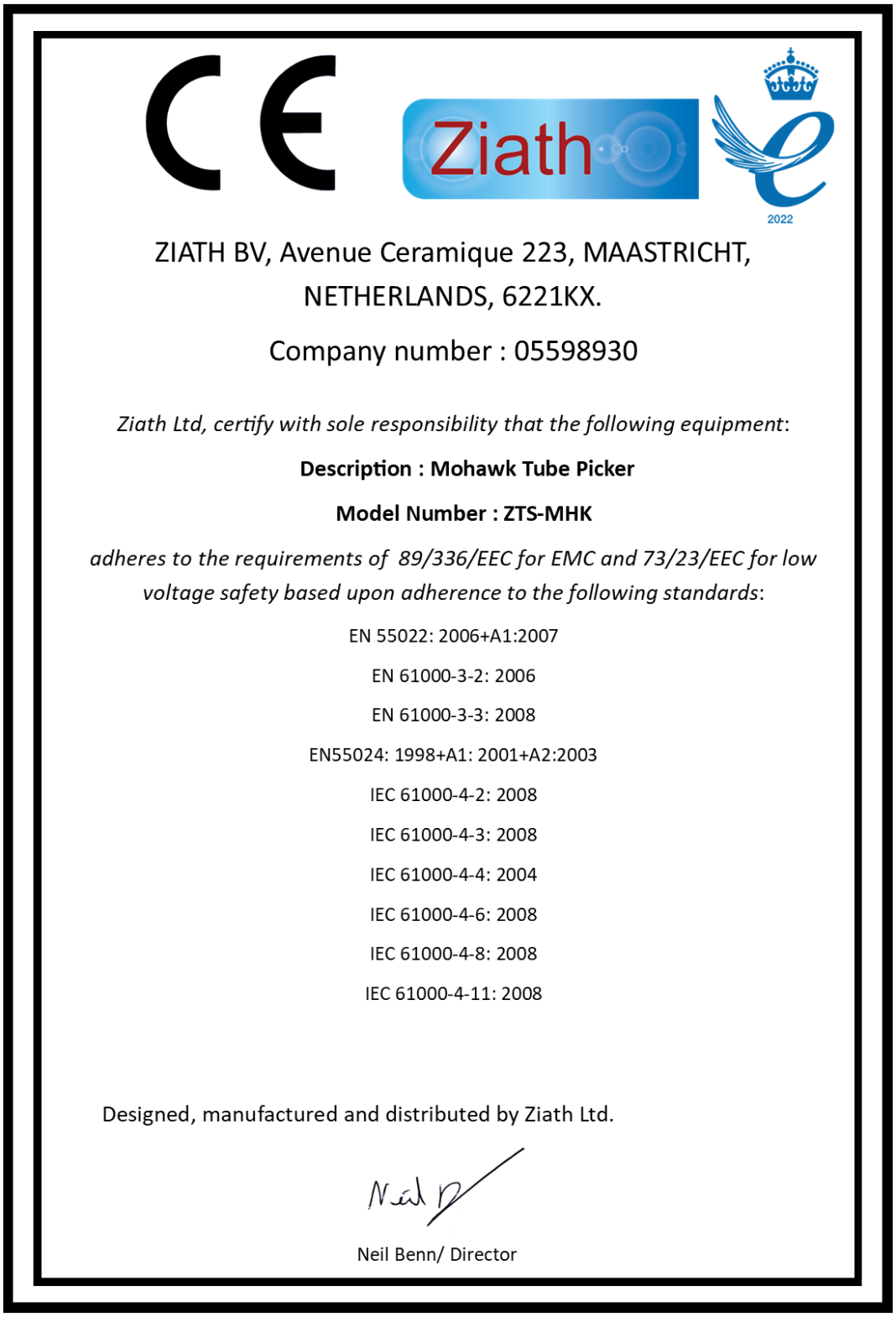
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="forceShutdown"></xs:element>
</xs:schema>
```

Appendix B

Errors

Beneath is a table of the possible errors that can be returned:

Error Code	Description
1	Server is busy
2	Not valid xml
3	Driver error
4	Lid is open
5	Force pins operation failed
6	Reset pins failed
7	Path not found
8	Device Communication error
9	Worklist validation error
10	Error generating worklist report
11	Error loading worklist
12	Worklist already loaded
12	Worklist is not loaded
14	Command not valid
15	Worklist is not finished
16	No linear reader configured



ZIATH BV, Avenue Ceramique 223, MAASTRICHT,
NETHERLANDS, 6221KX.

Company number : 05598930

Ziath Ltd, certify with sole responsibility that the following equipment:

Description : Mohawk Tube Picker

Model Number : ZTS-MHK

adheres to the requirements of 89/336/EEC for EMC and 73/23/EEC for low voltage safety based upon adherence to the following standards:

EN 55022: 2006+A1:2007

EN 61000-3-2: 2006

EN 61000-3-3: 2008

EN55024: 1998+A1: 2001+A2:2003

IEC 61000-4-2: 2008

IEC 61000-4-3: 2008

IEC 61000-4-4: 2004

IEC 61000-4-6: 2008

IEC 61000-4-8: 2008

IEC 61000-4-11: 2008

Designed, manufactured and distributed by Ziath Ltd.

Neil Benn/ Director

RoHS

ZIATH BV, Avenue Ceramique 223, MAASTRICHT,
NETHERLANDS, 6221KX.

Company number : 05598930

Ziath Ltd, certify with sole responsibility that the following equipment:

Description : Mohawk

Model Number : ZTS-MHK

adheres to the requirements of the directive on the restriction of the use of certain hazardous substances in electrical and electronic equipment (RoHS 2011/65/EU including amendment 2015/863/EU) and the harmonized standard EN IEC 63000:2018 and are verified as "RoHS compliant".

Designed, manufactured and distributed by Ziath Ltd.

Neil Benn/ Director

18th January 2021