**USIS Protocol for ESP32 RESTful Interface**

The **Universal Spectroscope Interface Specification (USIS) protocol** is designed to control any astronomical spectroscope for remote operation. This protocol was proposed by Shelyak Instruments company ([www.shelyak.com](http://www.shelyak.com/)) but only included a serial protocol. This document proposes a wireless protocol.

Using a standard protocol makes possible to any device with any software. It helps the developers (both hardware and software) with precise guidelines.

**Universal Spectroscope Interface Specification (USIS)**

A collaboration between Shelyak Instruments, the “Nice people”, and the amateur spectroscopy community

| Version | Date | Author | Revision |
| --- | --- | --- | --- |
| 1.0.0 | 22 Sep. 2022 | François COCHARD, Etienne Cochard, Nicolas Durand | First public version |
| 1.1.0 | 09 Sept 2024 | Gord Tulloch | Proposed REST wifi interace |
|  |  |  |  |

**Introduction**

The spectroscopy is nowadays in a great development, an in the last years more and more instruments are available on the market (or made in DIY mode). Among these instruments, the number that can be remotely controlled is increasing also. Since the remote control requires a piece of software, there is an increasing need for a specification of the interface between the hardware (the spectroscope, or some modules controlling it) and the software.

This is the purpose of this document to define such a specification.

We call this specification USIS for Universal Spectroscope Interface Specification.

This specification has two major goals :

* For the observers, the compliance of a spectroscope to USIS ensures that it can be controlled by their preferred software (if this software complies to USIS). On the other hand, choosing a USIS compliant software ensures that it will be able to control any USIS spectroscope.
* For the developers (hardware and software), this gives strong guidelines when creating new products.

The overall intention when writing this document is to avoid having as many interfaces and protocols as instruments, and contribute to the development of the spectroscopy by making the life of all actors much simpler.

General principles

The USIS is based on few principles:

* This specification is 'platform independent' : it can be translated to real applications on any OS (Windows, Linux or Mac) with no restriction.
* The communication between the PC and the spectrograph can be used through different ways. In the initial version we defined a serial/USB protocol and a API Rest (Ethernet) definition. Other ways may be added in the Future.
* The communication works at the highest possible level. For instance, the grating angle (if available) is given in degrees from the 0th order, and not in motor steps (this is the job of the spectroscope to convert into steps). All the informations must be as much generic as possible.
* The communication works in both directions. The computer is always the initiator of a request and the spectroscope – the controlled device – always replies to the computer.
* A USIS spectroscope or device can offer only part of the USIS commands. For instance, a calibration module can control the light sources, but not the grating properties. Also, a spectroscope can be controlled by different modules, each of them being USIS compliant.
* For any specific instrument, the developers can add some features (commands, for instance). In this case the commands will be ignored ; and the developers must ensure that even without using these commands the system will work properly. For instance, we can imagine a spectroscope that gets the humidity inside the instrument. In such a case, the compliance to USIS requires that it can work even if this features is not used.

**1. Properties**

A spectroscope offers a list of properties that can be read or controlled (written).

There are 4 groups of properties :

* Grating properties
* Slit properties
* Focus properties
* Light source properties

There are 3 types of properties :

* FLOAT, for any real parameter (ie focus position, grating angle...).
* ENUM, for lists of position (slit ID, Light source...). For instance, a light source can take several values : SKY, FLAT, CALIB, DARK.
* TEXT (string), for basic (and mostly static) information.

**Attributes**

For each type of property, we define several attributes. Some attributes are common to all property types, and some others are specific to each type.

Each attribute can be:

* RO READ only
* RW READ / WRITE

Common attributes for all property types:

* NAME: written with upper case, underscore (\_) or numbers. Max length is 25 characters. The first character indicates the group of properties it belongs. *Note:* a property cannot be named 'ALL', to let the possibility to send a general request to all properties.
* VALUE: the current value of the property (it can be a boolean, a string, an integer or a float value, following the property type)
* STATUS: gives the current status of the property. The status refers to the property, not to the attributes. Then one property has only one status. The principle is based on Indi properties status. It is usual to associate each status to a given color, as follows:

| **Status** | **Color** | **Description** |
| --- | --- | --- |
| N\_A | Gray | Not Applicable. The device is performing no action with respect to this property |
| OK | Green | Last operation performed on this property was successful and active |
| BUSY | Yellow | The property is performing an action |
| ALERT | Red | The property is in critical condition and needs immediate attention |

Note: ;and \* are reserved for the format of the requests and replies. They must NOT be used in the NAME or VALUE texts.

Specific attributes for a FLOAT property:

* MIN: minimal value (float)
* MAX: maximal value (float)
* UNIT: unit (string)
* PREC: precision (float). This attribute can be used to define if the target value is reached.

There is no specific attributes for an ENUM or TEXT property.

**2. Commands**

We define a set of commands to control any property of the device:

* GET: to get the current value of an attribute.
* SET: to change the value of an attribute (if it can be changed).
* STOP: to immediately stop any movement (in the case of a motorized feature, like the grating angle).
* INFO: to get general information from a property (type, attribute, ENUM values, Read/Write capability)
* CALIB: to calibrate the property. It gives an actual value for the current position.
* FACTORY\_RESET: to reset the attribute values to factory values (if any).
* SYSTEM: command reserved for maintenance. You can use it for your own hidden functions.

The software (PC) is always the initiator of the communication.

When a command is sent by the PC the command must be send in less than a timeout. The spectroscope/device replies in a maximal time frame (this allows a time out detection).

This is an important point: if the command takes some time to be executed (like a motor movement, for instance), the device must immediately reply to the PC, not waiting for the completion of the request.

The reply will contain the status of the property ; if a motor is still moving, the status is BUSY, and this is the responsibility of the PC to check when the request is completed, by requesting repeatedly the status of the property (wait 50 ms between calls).

Requests and Reply structures

A request can be sent by the computer to the device with following format

COMMAND;PROPERTY;ATTRIBUTE; VALUE (if any)

And the reply from the device has the following general structure:

ERROR\_CODE;PROPERTY;ATTRIBUTE;STATUS;VALUE

The actual details of the structure depends on each command.

There are two types for the ERROR\_CODE.

* The first type is a Communication error, if the request has not been properly received.
* The second type is a Message error and depends on the protocol. It will be detailed below section 'protocols'. In this case, the request has been properly received by the device, but cannot be executed, for any reason.

for a complete list of errors, see. [Communication errors](Communication errors)

3. Communication Protocol

The communication between the PC and the device (spectroscope) can be made in various ways.

In a first time, we consider that the communication is made through a serial port. A second option through Rest API, for instance (using a web browser) may be defined in a next revision.

Serial communication

All messages are sent through a serial port.

For serial the communication parameters are by default:

* 9600 bauds
* No parity
* One Stop bit
* Byte size = 8 bits

Direct USB serial do not require special configuration.

Timeout: The message must be fully received in less than 200ms, the response must be received in less than 300ms.

The general format of a request from PC to the device is as follows:

COMMAND;PARAMETER;ATTRIBUTE;VALUE\*CHECKSUM \n

* The maximum length of a message is 150 characters.
* Elements are separated by a ";" (semi colon)
* Checksum is separated from Elements by "\*" (star)
* The whole command is terminated by a new line "\n"
* All elements are ASCII, upper case.
* Floating point values are in the form -xxxx.yy (no thousand group, decimal point '.', no exponent, no '+' sign for positive values, '-' for negatives values)

Elements:

* COMMAND is the command name
* one of GET, SET, INFO, STOP, CALIB, FACTORY\_RESET
* PROPERTY is the Property name
* ATTRIBUTE is the attribute name (optional)
* VALUE is the value (optional)
* CHECKSUM (optional)

*Optional elements:* You can avoid sending optional values when the message do not require them. You can avoid sending the checksum element, in this case, do not set the checksum separator. *When you do not send checksum, response do not include checksum.*

When an request is received by the device, it replies immediately with an acknowledgment message with following format:

in case of error:

ERROR;ERROR\_DESCRIPTION\*CHECKSUM\n

where:

* ERROR is the error code Cxx or Mxx
* ERROR\_DESCRIPTION is a human readable explanation of the error (in english)
* CHECKSUM is a checksum (2 hexadecimal uppercase digits exactly). The checksum is only present if the request has a checksum.

in case of success:

M00;PROPERTY;ATTRIBUTE;STATUS;VALUE\*CHECKSUM\n

where :

* M00 is the error code 'NO ERROR'
* PROPERTY is the Property name,
* ATTRIBUTE is the attribute name,
* STATUS is the Status (cf. status codes)
* VALUE is the current value of the property
* CHECKSUM is a checksum (2 hexadecimal uppercase digits exactly). The checksum is only present if the request has a checksum.

Checksum

Checksum is optionnal. If you use checksum, here is how it is computed:

The Python code to calculate the checksum

def checksum(str):

cks = 0

stringByte = bytes(str,'ascii')

for car in stringByte :

cks = cks ^ car

return cks

The same code in C++ (Arduino, for instance) is :

unsigned char protocol\_calcChecksum( char\* message, size\_t msg\_length )

{

size\_t i;

unsigned char check = 0;

for (i = 0 ; i < msg\_length ; i++ )

check = check ^ message[i];

return check;

}

When present, checksum is always prefixed by a checksum separator: '\*' Its value is in hexadecimal uppercase.

[..message..]\*A9\n

Commands

Let's look more in detail to each command, through real examples. *For clarity, we do not include CHECKSUM nor ending "\n"*

Command GET

This is the main command to get the value of any attribute of any property.

To get the minimal value for a PROPERTY, we use the following command:

GET;PROPERTY;ATTRIBUTE

and the device replies:

M00;PROPERTY;ATTRIBUTE;STATUS;(value)

M00 means that there is no error,

STATUS means that the status of the property is OK,,

To get the actual current grating angle, the request is:

GET;GRATING\_ANGLE;VALUE

and the device replies, for instance:

M00;GRATING\_ANGLE;VALUE;BUSY;12.33

In this case, BUSY means that the grating is still rotating. You should request the same command several times until the grating angle reaches its target. Then, the status will become OK.

Command SET

This is the main command to set (change) the value of a property attribute.

SET;PROPERTY;VALUE;(target value)

For instance, to rotate the grating up to the angle 45.3°, use the following command:

SET;GRATING\_ANGLE;VALUE;45.3

and the device replies:

M00;GRATING\_ANGLE;VALUE;BUSY;19.38

Once the grating has reach the target angle, the reply to the same command is:

M00;GRATING\_ANGLE;VALUE;OK;45.27

Note that the actual value can be slightly different from the initial request, because of the technology used (for instance it can be here a stepping motor), but the difference between target and actual position is always less than |Target - actual|.

This can be tested by checking the PREC (precision) attribute of the property.

If you send a request to change a value that is Read Only (RO), then the device returns an error message (refer to error codes).

You can send a SET request even if the Property is BUSY. In this case, the new request replaces the on-going one (ex: if the grating angle is moving to a given target, you can send a request to move to another target).

Command STOP

This command is an emergency action, if there is a problem with a motor. The usage is:

STOP;GRATING\_ANGLE

The device replies:

M00;GRATING\_ANGLE;OK;28.6

The STOP command can be used to stop all devices in a single operation:

STOP;ALL

In this case, the reply should be

M00;STOP;ALL;OK

Command INFO

This is used to get details about a given property.

In this case, no attribute is given:

INFO;GRATING\_ANGLE

The device replies, for instance:

M00;GRATING\_ANGLE;FLOAT;DEGREE;0.1

Which means that this is a property of FLOAT type, the unit is DEGREE, and the precision is 0.1°.

Command CALIB

This command gives a value to the current position of a float property. This is the way to calibrate a property.

CALIB;GRATING\_ANGLE;32.21

The device replies, for instance (this is the same format as GET command):

M00;GRATING\_ANGLE;VALUE;OK;32.21

Command FACTORY\_RESET

This command resets all the attributes of a given property with original (factory) values.

FACTORY\_RESET;GRATING\_ANGLE

The device replies the same message as 'info' command, for instance:

M00;GRATING\_ANGLE;FLOAT;DEGREE;0.1

Which means that this is a property of FLOAT type, the unit is DEGREE, and the precision is 0.1°.

Command SYSTEM

This command allows to create maintenance functions. There is no specific documentation.

Errors

If a problem occurs during the communication (the message does not comply to the USIS protocol), the device returns an error message with following format :

Cxx;COMMUNICATION\_ERROR\*HH

where :

Cxx is the error code (C stands for Communication)

COMMUNICATION\_ERROR is the error message (text)

HH is the checksum (mandatory for a communication error)

The Communication error codes can be one of the following:

| Code | Code | Description |
| --- | --- | --- |
| C01 | TIMEOUT | Timeout reached (communication lost?) |
| C02 | BAD REQUEST | Unexpected end of the message |
| C03 | BAD CHECKSUM | Bad checksum |
| C04 | OVERFLOW | Message is too long (exceeds the buffer size) |

The Message error codes can be one of the following:

| Code |  | Description |
| --- | --- | --- |
| M00 |  | OK, No error |
| M01 | UNKNOWN PROPERTY | Unknown property |
| M02 | UNKNOWN ATTRIBUTE | Unknown attribute |
| M03 | READONLY | The value is read only |
| M04 | BAD VALUE TYPE | Bad value type you try to set a value that is not accepted ie. string into a float or float into an int |
| M05 | NO VALUE GIVEN | No value given you forgot to set a value |
| M06 | UNKNOWN COMMAND | Unknown command |
| M07 | OUT OF RANGE | Out of Range |
| M08 | BAD VALUE | Bad value (check enum value) |
| M09 | BAD INDEX | Bad index |
| M10 | NO POWER | No power to execute requested action |

Introspection

Usis enables a command for introspection. A caller can enumerate all properties and attributes.

Command is "INFO", followed by a REQUEST and one or more values ex. INFO;PROPERTY\_NAME;0

| REQUEST | Value 1 | Value 2 | Result |
| --- | --- | --- | --- |
| PROPERTY\_COUNT |  |  | INT number of properties |
| PROPERTY\_NAME | property index |  | TEXT property name |
| PROPERTY\_TYPE | property index |  | TEXT property type, one of TEXT, ENUM, FLOAT, INT |
| PROPERTY\_STATE | property index |  | TEXT property state, one of OK, BUSY, ALERT, IDLE, NA |
| PROPERTY\_ATTR\_COUNT | property index |  | INT number of attributes for the property |
| PROPERTY\_ATTR\_NAME | property index | attribute index | TEXT attribute name |
| PROPERTY\_ATTR\_MODE | property index | attribute index | RW read write RO read only |
| PROPERTY\_ATTR\_ENUM\_COUNT | property index | attribute index | INT number of enum values for the attribute |
| PROPERTY\_ATTR\_ENUM\_VALUE | property index | enum index | TEXT enum value |

PROPERTY\_ATTR\_ENUM\_VALUE has a special attribute index / enum index. ex: INFO;PROPERTY\_ATTR\_ENUM\_VALUE;0;2 means property 0, attribute 0 (implicit), enum 2

Properties List

| Name | Description | Type |
| --- | --- | --- |
| Device Properties |  |  |
| DEVICE\_NAME | Name of the device | TEXT |
| SOFTWARE\_VERSION | Returns the software version | TEXT |
| PROTOCOL\_VERSION | Returns the protocol version | TEXT |
| TEMPERATURE | Temperature in the spectroscope (°C) | TEXT |
| HUMIDITY | Humidity in the spectroscope (%) | TEXT |
| Grating Properties |  |  |
| GRATING\_ID | Current Grating ID (position), if the system can change the grating. Can be used to change the grating. | ENUM |
| GRATING\_ANGLE | Gives the current grating angle, in degrees vs order 0. This function (and the next ones) is used if the grating position is driven by angle. enum values: 600,300,150,1200,1800 | FLOAT |
| GRATING\_WAVELENGTH | Gives the current grating wavelength, in nm.This function (and the next ones) is used if the grating position is driven by wavelength. | FLOAT |
| GRATING\_DENSITY | Gives the current grating dentity, in LINE/MM. | FLOAT |
| Slit Properties |  |  |
| SLIT\_ID | Slit position | ENUM |
| SLIT\_WIDTH | Slit width | FLOAT |
| SLIT\_ANGLE | Slit Angle | FLOAT |
| Focus properties |  |  |
| FOCUS\_POSITION | Focus position | FLOAT |
| Light Source properties |  |  |
| LIGHT\_SOURCE | Current light source | ENUM |
|  |  |  |

Conclusion

When following the above guidelines, developers will save significant time and energy, and observers will get simpler configuration of their setup.

Of course, this document will evolve over the time, when our common experience grows.

Don't hesitate to send your comments or requests to François Cochard at Shelyak Instruments ([francois.cochard@shelyak.com](mailto:francois.cochard@shelyak.com)).