SpecsLab Prodigy Remote Control Protocol

VERSION 1.5

NOVEMBER 13, 2014

SPECS™

Inhalt

1 Introduction

1.1 SpecsLab Prodigy Behavior

In remote control mode:

- Prodigy shows that it is being remotely controlled.
- Prodigy shows the data as it is acquired.
- Experiment Editor and Remote Control cannot acquire data at the same time.
- A remote acquisition can be paused or aborted within the Remote Control plugin.
- Remote Control does not interfere with running local acquisitions.
- The spectrum is recorded with a one or two-dimensional detector and is written in an m×n-array.
- During data acquisition, only one scan will be recorded. Multiple scans are taken by repeating the acquisition. Data accumulation and / or averaging as well as storage is performed by the remote client.
- Local device parameters will not be set remotely.

1.2 General Protocol Description

- Communication is on a request / reply basis.
- In this context, SpecsLab Prodigy acts as the server.
- Requests are sent from the client application and are answered by SpecsLab Prodigy.
- SpecsLab Prodigy only accepts a single connection.
- The protocol format is plain ASCII text via TCP/IP.
- The TCP port of the remote control server is 7010.
- Each command is acknowledged (by “OK”, “OK: [...]” or “Error: <code> “message” ”).
- Error codes are 16 bit decimal positive integer values. Error codes are not unique but rather are an error “class”. Emphasis is put on the textual information.
- Each command may return an error: the error condition is given by an integer error code and a textual description (as a message string).
- Every command can potentially be answered with an error.
- Each command and response are terminated by a newline character “\n”.
- Token separation by <space> (ASCII 32dec).
- Message (character) strings are enclosed in double quotes: "<message>"; double quotes inside strings have to be escaped with a backslash (“\”).
- Requests start with “?” followed by a request ID.
- Responses start with “!” followed by the corresponding request ID.
- Request IDs have a fixed length of 4 hexadecimal digits (e.g. 0001, AB03) where requests and responses have matching IDs.
- All command and parameter names are case sensitive.

- In the first version, no binary transfer is supported.
- Commands which take a longer time to complete (for example, an acquisition) are performed asynchronously; a reply will be issued as a confirmation that the command will be / has been started and the actual state can be queried through other requests.
- Replies should normally be sent within one second; if a timeout occurs, the sender has to manage this depending on the command and state (resend, abort, ...).
- No automatic error mechanism is specified with this protocol.

1.3 Request Syntax

```
?<id> Command [InParams]
```

where:

`id`	Unique request identifier (hexadecimal value, always 4 digits)
`Command`	Command name (character token, camel case, no spaces)
`InParams`	Optional list of input parameters (“key:value”-list, space separated), specific for each command; the order of parameters is arbitrary.

EXAMPLES:

```
?0107 Connect
?0231 GetAnalyzerParameterInfo ParameterName:“DetectorVoltage”
?010B DefineSpectrum StartEnergy:1.0 EndEnergy:20.0 StepWidth:1.0
[...]
?010C Disconnect
```

1.4 Response Syntax

```
!<id>        OK
```
or
```
!<id>        OK: [OutParams]
```
or
```
!<id>        Error: <Code> [Reason]
```

where:

`id`	Is the id of the corresponding request (4 digits, hexadecimal)
`OutParams`	List of output parameters (“key:value” list, space separated) or error code and textual error message
`Code`	Decimal representation of the error (see section 5).
`Reason`	Textual description of the error

EXAMPLES:

```
!0028 OK
!0028 OK: DetectorVoltage:1950.0
!0198 Error: 201 “Start energy should be above …”
!0029 OK: ControllerStatus:running EnergyPosition:230.3
```

2 List of Commands (Requests from Client to SpecsLab Prodigy)

Every request can potentially be answered with an error reply.

2.1 Connect

Open connection to SpecsLab Prodigy.

Parameters: (None)
Response: `OK: ServerName:<Text> ProtocolVersion:<Major.Minor>`

`Text`	Arbitrary string reported from SpecsLab Prodigy
`Major`	Major number of the supported protocol version
`Minor`	Minor number of the supported protocol version

```
EXAMPLE:
?0100 Connect
!0100 OK: ServerName:“SpecsLab Prodigy 4.0” ProtocolVersion:1.2
```

2.2 Disconnect

Close connection to SpecsLab Prodigy.

Parameters: (None)
Response: `OK`

```
EXAMPLE:
?00A0 Disconnect
!00A0 OK
```

2.3 DefineSpectrumFAT

Send FAT spectrum specification for subsequent acquisition. Existing data must be cleared first.

Parameters:

`StartEnergy`	First point in spectrum in eV
`EndEnergy`	Last point in spectrum in eV
`StepWidth`	Delta between measurement points in eV
`DwellTime`	Dwell time of the detector in seconds
`PassEnergy`	Pass energy in eV
`LensMode`	Lens mode (as string)
`ScanRange`	HSA voltage range for scanning (as string)

Response: `OK`

```
EXAMPLE:
?0101 DefineSpectrumFAT StartEnergy:300.0 EndEnergy:320.0
      StepWidth:0.01 DwellTime:0.1 PassEnergy:10.0
      LensMode:"MediumArea" ScanRange:"1.5kV"
!0101 OK
```

2.4 DefineSpectrumSFAT

Send SFAT spectrum (snapshot) specification for subsequent acquisition. Existing data must be cleared first. Note: Step width and pass energy are computed automatically wrt the current detector calibration.

Parameters:

`StartEnergy`	First point in spectrum in eV
`EndEnergy`	Last point in spectrum in eV
`Samples`	Number of acquisition samples
`DwellTime`	Dwell time of the detector in seconds
`LensMode`	Lens mode (as string)
`ScanRange`	HSA voltage range for scanning (as string)

Response: `OK`

```
EXAMPLE:
?0101 DefineSpectrumSFAT StartEnergy:300.0 EndEnergy:320.0
      Samples:1 DwellTime:0.1 LensMode:"MediumArea" ScanRange:"1.5kV"
!0101 OK
```

2.5 DefineSpectrumFRR

Send FRR spectrum specification for subsequent acquisition. Existing data must be cleared first.

Parameters:

`StartEnergy`	First point in spectrum in eV
`EndEnergy`	Last point in spectrum in eV
`StepWidth`	Delta between measurement points in eV
`DwellTime`	Dwell time of the detector in seconds
`RetardingRatio`	Retarding Ratio
`LensMode`	Lens mode (as string)
`ScanRange`	HSA voltage range for scanning (as string)

Response: OK

```
EXAMPLE:
?0101 DefineSpectrumFRR StartEnergy:300.0 EndEnergy:320.0
      StepWidth:0.01 DwellTime:0.1 RestardingRatio:10.0
      LensMode:"MediumArea" ScanRange:"1.5kV"
!0101 OK
```

2.6 DefineSpectrumFE

Send FE spectrum specification for subsequent acquisition. Existing data must be cleared first.

Parameters:

`KinEnergy`	Kinetic Energy in eV
`Samples`	Number of acquisition samples
`DwellTime`	Dwell time of the detector in seconds
`PassEnergy`	Pass energy in eV
`LensMode`	Lens mode (as string)
`ScanRange`	HSA voltage range for scanning (as string)

Response: OK

```
EXAMPLE:
?0101 DefineSpectrumFE KinEnergy:300.0 Samples:5 DwellTime:0.1
      PassEnergy:10.0 LensMode:"MediumArea" ScanRange:"1.5kV"
!0101 OK
```

2.7 ValidateSpectrum

Validate parameters defined by previous `DefineSpectrum<Type>` command. Existing data must be cleared first.

Parameters: (None)
Response: `OK: [OutParams]`

`OutParams`	"`key:value`" list of the actual parameter values of the spectrum command (potentially modified during the validation).

```
EXAMPLE:
?0102 ValidateSpectrum
!0102 OK: StartEnergy:300.0 EndEnergy:320.0 StepWidth:0.01
      Samples:2001 DwellTime:0.1 PassEnergy:10.0
      LensMode:"MediumArea" ScanRange:"1.5kV"
```

2.8 Start

Start data acquisition. Spectrum must have been validated first. An acquired spectrum remains valid when it is cleared.

Parameters: (None)
Response: `OK`

```
EXAMPLE:
?0102 Start
!0102 OK
```

2.9 Pause

Pause data acquisition.

Parameters: (None)
Response: `OK`

```
EXAMPLE:
?0102 Pause
!0102 OK
```

2.10 Resume

Resume a paused data acquisition.

Parameters:	(None)
Response:	`OK`

```
EXAMPLE:
?0102 Resume
!0102 OK
```

2.11 Abort

Abort a running or paused data acquisition.

Parameters:	(None)
Response:	`OK`

```
EXAMPLE:
?0102 Abort
!0102 OK
```

2.12 GetAcquisitionStatus

Reports information about the status and the progress of the acquisition.

Parameters: (None)
Response: `Ok: ControllerState:<ContState> NumberOfAcquirePoints:<NumPts> [optional: Message:<Text> Details:<Text>]`

`ContState:`	`idle`	No spectrum is specified or a spectrum is not validated
	`validated`	Spectrum has successfully been validated
	`running`	Acquisition is running
	`paused`	Acquisition has been paused
	`finished`	Acquisition is finished (or has been aborted) and spectrum has not been cleared

`NumberOfAcquirePoints`	positive integer value
`Message`	Error Message
`Details`	Error Details

```
EXAMPLES:
?0102 GetAcquisitionStatus
!0102 OK: ControllerState:idle

?0102 GetAcquisitionStatus
!0102 OK: ControllerState:validated

?0102 GetAcquisitionStatus
!0102 OK: ControllerState:running NumberOfAcquiredPoints:12

?0103 GetAcquisitionStatus
!0103 OK: ControllerState:paused NumberOfAcquiredPoints:75

?0103 GetAcquisitionStatus
!0103 OK: ControllerState:finished NumberOfAcquiredPoints:92
```

2.13 GetAcquisitionData

Request a slice of data from the acquisition buffer. The buffer is not modified through this reading (non-destructive). Reading from parts of the buffer which have not been acquired is not an error and returns zeros.

Parameters:

`FromIndex`	Index of first point to be reported	($0 \leq i <$ number energy channels)
`ToIndex`	Index of last point to be reported	($0 \leq i <$ number energy channels)
Response:	`OK: Data:[Values]`	

`Values` List of double values which have to interpreted as a two-dimensional data set (non-energy channels) × (sample) of the form

[s_1i, ..., s_1j, s_2i, ..., s_2j, ..., s_Mi, ..., s_Mj]

where M equals the number of non-energy channels.

```
EXAMPLES:
?0102 GetAcquisitionData FromIndex:2 ToIndex:4
!0102 OK: Data:[247599,246218,240558,233324,230841,230169, …]
```

2.14 ClearSpectrum

If controller is in state finished, the command clears the internal spectrum buffer and sets the controller state to idle. During an acquisition an error is reported. A cleared spectrum remains valid until a new definition is send.

Parameters: (None)
Response: `OK`

```
EXAMPLE:
?0102 Clear
!0102 OK
```

2.15 GetAllAnalyzerParameterNames

Request all analyzer device parameter names.

Parameters:	(None)
Response:	`OK: ParameterNames:[Names]`

`Names`	List of parameter names

```
EXAMPLE:
?0231 GetAllAnalyzerParameterNames
!0231 OK: ParameterNames:["DetectorVoltage","Kinetic Energy Base", …]
```

2.16 GetAnalyzerParameterInfo

Request information about a single analyzer parameter.

Parameters:

`ParameterName`	Name of the parameter whose information is queried

Response: `OK: Type:<Type> ValueType:<ValueType> Unit:<Unit> [optional: Min:<Min> Max:<Max> Values:[…]]`

`Type`	LogicalVoltage or Setting (additional types must be further specified)
`ValueType`	Double, Integer, String (additional types must be further specified)
`Unit`	currently not supported (an empty string is returned)
`Min`	currently not supported
`Max`	currently not supported
`Values`	currently not supported

```
EXAMPLE:
?0231 GetAnalyzerParameterInfo ParameterName:"DetectorVoltage"
!0231 OK: Type:LogicalVoltage ValueType:double Unit:"V" Min:0.0 Max
3000.0

?0231 GetAnalyzerParameterInfo ParameterName:"AnalyzerStandbyDelay"
!0231 OK: Type:Setting ValueType:double Unit:"s" Min:0.0 Max
1000000000.0
```

```
?0231 GetAnalyzerParameterInfo ParameterName:"LensMode"
!0231 OK: Type:Setting ValueType:string Values:["MediumArea",
"LargeArea"]

?0231 GetAnalyzerParameterInfo "ScanRange"
!0231 OK: Type:Setting ValueType:string Values:["100V", "400V",
"1.5kV", "3.5kV"]
```

2.17 GetAnalyzerVisibleName

Request the analyzer device visible name.

Parameters: (None)
Response: `OK: AnalyzerVisibleName:Name`

`Name`	The analyzer device visible name.

EXAMPLE:

```
?0231 GeAnalyzerVisibleName
!0231 OK: AnalyzerVisibleName:”Phoibos HSA3500 150 R7 NAP”
```

2.18 GetAnalyzerParameterValue

Request the value of a single analyzer parameter.

Parameters:

`ParameterName`	Name of the parameter which is queried

Response: `Ok: <ParameterName>:<ParameterValue>`

`ParameterName`	Name of the reported parameter setting
`ParameterValue`	Value of the queried parameter

EXAMPLE:

```
?0231 GetAnalyzerParameterValue ParameterName:"DetectorVoltage"
!0231 OK: DetectorVoltage:1850.0
```

2.19 SetAnalyzerParameterValue

Sets the value of a single analyzer parameter.
Currently only values of parameter voltages can be set. This can be done at any time with no restrictions to requested acquisitions. Setting the voltage is active immediately.

Parameters:

`ParameterName`	Name of the analyzer parameter
`Value`	Value (parameter voltage in eV)

Response: `Ok`

```
EXAMPLE:
?0231 SetAnalyzerParameterValue ParameterName:"Kinetic Energy Base"
      Value:10.0
!0231 OK
```

2.20 DisconnectAnalyzer

Request to disconnect analyzer.

Parameters: (None)
Response: `Ok`

```
EXAMPLE:
?0231 DisconnectAnalyzer
!0231 OK
```

3 Remote Session Example

The TCP port of the remote control server is 7010.

```
?0001 Connect
!0001 OK: ServerName:"SpecsLab Prodigy 4.8-r44312"
ProtocolVersion:1.4

?0002 GetAllAnalyzerParameterNames
!0002 OK: ParameterNames:["NumEnergyChannels","Screen Voltage","Bias
Voltage Electrons","Bias Voltage Ions","Detector Voltage","Focus
Displacement 1","Maximum Count Rate [kcps]","Analyzer Standby Delay
[s]","Skip Delay Up/Down"]

?0003 GetAnalyzerParameterInfo ParameterName:"Screen Voltage"
!0003 OK: Type:LogicalVoltage ValueType:double Unit:""

?0004 GetAnalyzerParameterValue ParameterName:"Screen Voltage"
!0004 OK: Name:"Screen Voltage" Value:0

?0005 ValidateSpectrum
!0005 Error: 202 Remote Control: Validation failed.
No lens mode specified.
Please select a valid lens mode for spectrum 'Remote Control'.

?0006 DefineSpectrumFAT StartEnergy:300.0 EndEnergy:1500.0
StepWidth:1 DwellTime:0.1 PassEnergy:10.0 LensMode:"MediumArea"
ScanRange:"1.5kV"
!0006 OK

?0007 ValidateSpectrum
!0007 OK: StartEnergy:300 EndEnergy:1500 StepWidth:1 DwellTime:0.1
PassEnergy:10 LensMode:"MediumArea" ScanRange:"1.5kV"

?0008 Start
!0008 OK

?0009 Pause
!0009 OK

?0010 Resume
```

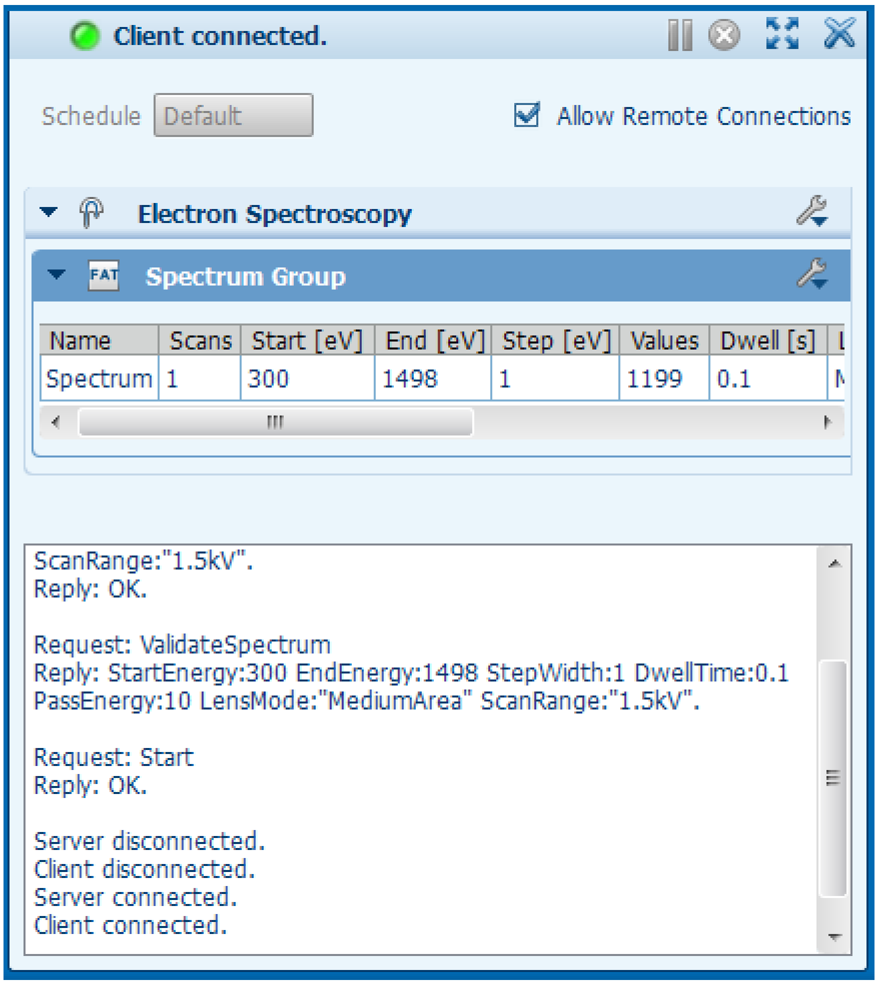
```
!0010 OK

?0011 GetAcquisitionStatus
!0011 OK: ControllerState:finished NumberOfAcquiredPoints:1194

?0012 GetAcquisitionData FromIndex:0 ToIndex:8
!0012 OK: Data:[247461,243729,239662,235772,232407,231056,232737,
238976,257519]

?0013 Disconnect
```

4 Plug-in Interface

In order to establish a remote connection:

1. Open the Remote Control View.
2. Select an experiment template from the Schedule-Selector.
3. Enable remote connections.
4. Communicate via TCP (see above).

During the remote session you can see the requests and replies in the lower part of the window.



Name	Scans	Start [eV]	End [eV]	Step [eV]	Values	Dwell [s]	L
Spectrum	1	300	1498	1	1199	0.1	M

```
ScanRange:"1.5kV".
Reply: OK.

Request: ValidateSpectrum
Reply: StartEnergy:300 EndEnergy:1498 StepWidth:1 DwellTime:0.1
PassEnergy:10 LensMode:"MediumArea" ScanRange:"1.5kV".

Request: Start
Reply: OK.

Server disconnected.
Client disconnected.
Server connected.
Client connected.
```

NOTE:
A new configuration can be specified with the Experiment Editor plug-in and must consist of a single spectrum definition in a single Electron Spectroscopy element. It is important that the configuration:

1. Contains a valid analyzer and source command, and
2. specifies a correct detector calibration plus analyzer slits.

All other parameters will be set via remote requests.

The default folder for configurations is <SPECS Settings Folder>\RemoteControl. An existing default configuration (Default.slt) will be loaded automatically when the plugin is opened.

5 List of Error Codes

Remote Control errors are primarily categorized by their layer:

- Connection Errors (Range 1 .. 99)
- Protocol Errors (Range 101 .. 199)
- Logical and Execution Errors (Range 201 .. 299)

The following gives an overview of the categories and the corresponding error codes.
A more detailed cause will be given in the textual error description of the response message (see section 1.4).

5.1 Connection Errors

These errors signal a failure at the message-passing level between client and server or a malformed incoming message.

Error	Reason
1	No server to connect to
2	Another client is already connected
3	Client is not connected
4	Malformed message

5.2 Protocol Errors

Protocol errors happen when a request has been forwarded to the server but is formatted incorrectly or the command arguments do not match the command's specification.

Error	Reason
101	Unknown command
102	Unknown error
103	Invalid argument sequence
104	Missing argument
105	Unknown argument
106	Invalid argument type

5.3 Logical and Execution Errors

Logical and execution errors occur when the command itself is syntactically correct but cannot be executed in the specified context. They may also arise on any other failure when the command is executed (e.g. a device error occurs).

Error	Reason
201	Failed to set the spectrum parameters
202	Validation error
203	Failed to start acquisition
204	Failed to clear spectrum
205	Failed to fetch parameter info
206	Unknown analyzer parameter
207	No data available
208	Invalid range
209	Currently acquiring spectrum
210	Spectrum contains data
211	Spectrum not validated
212	No running acquisition
213	Failed to disconnect analyzer
214	Trying to interfere with a running acquisition