Script Command Reference

A script command is a case insensitive keyword prefixed with a hyphen (-), backslash (\) or forward slash (/) and often followed by one or more parameters separated by spaces (any parameters containing spaces need to be enclosed in "double quotations"). Commands and keywords are case insensitive. All the available script commands are enumerated below including their abbreviations, shorter commands which sacrifice readability for speed/space, and their expected parameters. Each command is documented using the following syntax:

Script Command Syntax

Text to be typed as seen will be in **bold**, parameters for which you must supply a value will be enclosed in angle brackets < >, and any additional notes will be shown in gray/italicized text, for example:

-Command <Parameter 1 *this is the first parameter*> <Parameter 2 *this is the second parameter*>

Anything contained in square brackets [ ] is optional and sometimes includes a note in gray/italicized text clarifying when the option applies. Default values/behaviors for omitted optional parameter are included in the descriptive text for each script command. For example:

-Command <Parameter 1 *an integer*> [<Parameter 2> *only applies if Parameter 1 is even*]
If Parameter 2 is omitted, for an even value of Parameter 1, an error will be displayed and no action will be taken.

When optional parameters are enclosed in square brackets [ ] an ellipses ... will sometimes follow the closing bracket indicating that the optional parameter(s) can be repeated. If there are any specific instructions pertaining to when the parameter(s) can be repeated, or how many times, those will follow the ellipses ... as gray/italicized text. For example:

-Command <Optional Parameter Count *an integer*> [<Optional Parameter>]... *repeat number of times specified by Optional Parameter Count*

Parameters with a discrete set of options will contained in curly braces { } instead of angle brackets < > and the mutually exclusive options (only one can be used) will be separated by vertical bars |. The options will usually preceded by a brief description of what the options represent in gray/italicized text, for example:

-Command <Parameter 1> {*do something different* **True**|**False**}

Integral parameter value ranges will be separates by two periods .. and enclosed in angle brackets, for example <1..5> represents possible integer values of 1, 2, 3, 4, or 5. Floating point ranges will be separated by a hyphen instead, for example <0-5> represents any value between 0 and 5 including 0 and 5. If a range is included along with the name of the parameter the range will be in gray/italicized text, for example:

-Command <Parameter 1 *0..5*> <Parameter 2 *0-5*>

Running Script Commands from Prairie View

See the documentation for the Tools -> Scripts menu.

Running Script Commands Outside of Prairie View

There are three ways to run script commands from outside Prairie View, two of which can also be used from another computer on the same network, in order of ease of use those are:

- Script commands can be passed as **command line parameters** to Prairie View.exe. If Prairie View is already running any script commands passed as parameters will be sent to that running instance of Prairie View to be executed in the order they are received. If Prairie View is not already running, Prairie View will start up and any script commands included as parameters will be executed once Prairie View has finished loading. There are some special cases such as the -ConfigurationFile (-cf) script command which are processed as Prairie View is starting up and ignored otherwise
- For third party applications the **PrairieLink** API can be used for a variety of interactions with Prairie View including sending script commands
- Behind the scenes all script command sent from outside the running instance of Prairie View are send over a **TCP/IP** connection on port 1236. While PrairieLink is still preferred, it it possible to connect directly to Prairie View by opening a TCP over IP connection to port 1236 on the PC running Prairie View.

  When sending commands over TCP/IP an ASCII character 1 (not the number 1 which is ASCII character 49) should be used instead of spaces between commands and parameters. Also if a password is enabled (default) on the Edit Scripts dialog that password must be the first thing sent to Prairie View or the connection will be terminated immediately.

To indicate to Prairie View a sequence of commands is ready to execute, send a carriage return (ASCII character 13) followed by the line feed (ASCII character 10). After Prairie View receives the commands and arguments followed by this terminator it will respond with ACK also terminated by a carriage return and line feed character, then proceed to execute the commands. Once the commands have been executed Prairie View will respond DONE again terminated by a carriage return and line feed character.

Any commands which return a result, such as -GetState (-gts), will send a response, again terminated by a carriage return and line feed character, after the ACK response, but before the DONE response.

When finished with the connection send the -Exit (-x) command to gracefully close the connection with Prairie View, failure to do so will result in an exception being thrown in Prairie View when the connection is terminated.

Command Categories

(* denotes a TCP/IP only command)

General Commands
-AppendNote (-an)
-ClearNotes (-cn)
-ConfigurationFile (-cf)
-DoNotWaitForScans (-dw)
-ExecuteProgram (-ep)
-ExecuteScript (-es)
-Exit (-x)*
-Help (-?)
-MessageToOperator (-mto)
-NoWait (-nw)
-Shutdown (-xsd)
-Silent (-s)
-UtilityButton (-ub)
-Wait (-wt)
-WaitForInputTrigger (-wfit)

File Commands
-LoadEnvironment (-le)
-LoadTemplate (-lt)
-LoadImages (-li)
-LoadMarkPoints (-lmp)
-LoadROIFile (-lrf)
-LoadVoltageOutput (-lvo)
-SaveEnvironment (-se)
-SetSavePath (-p)
-SetFileIteration (-fi)
-SetFileName (-fn)

Action Commands
-ClearBOTs (-cb)
-ClearROIs (-cr)
-GetBOTRegions (-gb)
-GetROIs (-gr)
-SetActionAfterFrame (-af)
-SetActionAfterScan (-as)

Image Commands
-ImageWindowFit (-iwf)
-ImageWindowLarger (-iwl)
-ImageWindowOriginal (-iwo)
-ImageWindowSmaller (-iws)

Visual Script Commands
-VisualScriptForm (-vsf)
-VisualScriptLabel (-vsl)
-VisualScriptControl (-vsc)
-VisualScriptButton (-vsb)
-VisualScriptNewLine (-vsn)
-VisualScriptShow (-vss)

Acquisition Commands
-Abort (-stop)*
-DroppedData (-dd)*
-GetImage (-gi)*
-LimitGSDMABufferSize (-lbs)
-LiveScan (-lv)
-MarkAllPoints (-slm)
-MarkPoints (-mp)
-MarkPointsMetadata (-mpm)
-PointScan (-ps)
-ReadRawDataStream (-rrd)*
-SetAcquisitionMode (-sam)
-SingleScan (-ss)
-SingleScanTriggered (-sst)
-StreamRawData (-srd)*
-TSeries (-ts)
-TSeriesLoad (-tsl)
-WaitForScan (-w)

Line Scan Commands
-GetFreehandLine (-gl)
-LineScan (-ls)
-LineScanDialog (-ld)
-LineScanLines (-lsl)
-LineScanMode (-lm)
-LoadLineScan (-lls)

Z-Series Commands
-FindSlice (-fs)
-SetZSeriesDisplay (-zsd)
-SetZSeriesNumberOfSlices (-zsn)
-SetZSeriesStart (-zsb)
-SetZSeriesStepSize (-zsz)
-SetZSeriesStop (-zse)
-ZSeries (-zs)
-ZSeriesAddSlice (-zsas)
-ZSeriesClear (-zscl)
-ZSeriesInsertSlice (-zsis)
-ZSeriesLoad (-zsl)
-ZSeriesMoveTo (-zsmt)
-ZSeriesRemoveSlice (-zsrs)
-ZSeriesSave (-zss)
-ZSeriesStepMode (-zssm)

Scan Setting Commands
-EnterROI (-er)
-GetState (-gts)*
-ParameterSet (-pa)
-ROILoad (-roi)
-SamplesPerPixel (-spp)*
-SetChannel (-c)
-SetDwellTime (-dt)
-SetFrameAveraging (-fa)
-SetImageSize (-is)
-SetOpticalZoom (-oz)
-SetScanRotation (-sr)
-SetState (-sts)
-SetCustomOutput (-co)

Hardware Commands
-Camera (-ca)
-CenterGalvos (-cg)
-GetETLValue (-ge)*
-GetMotorPosition (-gmp)*
-MoveMotor (-mr)
-NikonNiMicroscope (-nni)
-NikonTiMicroscope (-nti)
-PanGalvo (-png)
-SendGPIOCommand (-gc)
-SendMAMCCommand (-mc)
-SendPiezoCommand (-pc)
-SendServoCommand (-sc)
-SendResonantCommand (-rc)
-SetMotorPosition (-ma)
-SetObjectiveLens (-sol)
-SFC (-sfc)
-SLMCalibrationMask (-scm)
-ZDeviceControl (-zdc)
-ZeissMicroscope (-zm)

Laser/PMT Commands
-AlternateBeamRoute (-abr)
-SecondaryLaserBeamRoute (-slbr)
-SetLaserPower (-lp)
-SetMultiPhotonWavelength (-mpw)
-SetPMTGain (-pg)

Shutter Commands
-OverrideHardShutter (-ohs)
-SetHardShutter (-hrd)
-SetSoftShutter (-sft)

Stage Position Commands
-AddStagePosition (-spa)
-ClearStagePositions (-spc)

General Commands

{**-AppendNote**|**-an**} <Note>
Adds the given text to the notes. Notes are accessible under the Applications menu on the main form or by pressing F8. Notes containing specific keywords/tokens enclosed in angle brackets < > will be replaced with actual values when the script command is executed. Keywords are case-insensitive and include any of the xml keys used to record Prairie View's state (these include <pixelsPerLine>, <rotation> and <opticalZoom> to name a few) as well as specific keywords such as: <time> which translates to the current time and <lineScanCount> which translates to the current line number when appending to a line scan.
Back to Commands

{**-ClearNotes**|**-cn**}
Clears any notes currently recorded. Notes are accessible under the Applications menu on the main form or by pressing F8.
Back to Commands

{**-ConfigurationFile**|**-cf**} <Filename **.xml path optional*>
This command has no effect unless Prairie View is launched with it (i.e. it is included as part of the Prairie View shortcut). This command specifies another configuration file to load instead of the default 'C:\ProgramData\Bruker Fluorescence Microscopy\Prairie View\x.x.x.x\Configuration\configuration.xml'. If just a filename is provided with no path, Prairie View will assume the file is in the same folder as the default configuration file (which is where it should be to continue working after a software upgrade). This allows a single version of Prairie View to support two separate hardware configurations (i.e. one with 2P laser control and one without).
Back to Commands

{**-DoNotWaitForScans**|**-dw**} [{*execute commands during scans* **True**|**False**}]
When passing either **True** or no parameters, script commands will no longer wait for a scan in progress to complete before executing, with the exception of commands which start a new scan, which will always wait. T-Series and Live scans are exempt from this command, any commands, which do not start a new scan, sent during a T-Series or Live scan will never wait regardless of how DoNotWaitForScans is called. Passing **False** will resume the default behavior of waiting for any scans in progress to complete prior to executing script commands.
Back to Commands

{**-ExecuteProgram**|**-ep**} <Filename *including path*> [<Argument>]... *include any arguments to be passed to the program being executed*
Runs the program specified by the given filename and passes along any additional arguments provided. This command can be used to launch another program automatically as Prairie View starts up.
Back to Commands

{**-ExecuteScript**|**-es**} <Category> <Script> [{*Iterations* <Integer>|**allXY**|**AllROI**|**allZSlices**} [<Iteration Interval *seconds*> [<Iteration Variable>]]]
Executes the specified script within the specified script category. If the script being started calls any script before it (circular reference), a warning message will be displayed and the entire script will be terminated. The optional argument, iterations, indicates the number of times that the specified script should be repeated. If this value is an integer (1, 2, 3, ...) that indicates the number of times to execute the specified script. If the value of iterations is **allXY**, that indicates to execute the specified script at all of the defined XY stage locations. If the value of iterations is **allROI**, that indicates to execute the specified script for all of the ROI definitions. If the value of iterations is **allZSlices**, that indicates to execute the specified script at each slice of the currently defined Z-series. The optional argument, iteration interval, indicates the amount of time (in seconds) to wait from the start of one iteration to the start of the next iteration of the specified script. If the value for iterations is **allXY**, then the value of iteration interval indicates the amount of time to wait between executing the specified script at each of the XY stage locations. If the value for iterations is **allROI**, then the value of iteration interval indicates the amount of time to wait between executing the specified script for each ROI definition. If the value for iterations is **allZSlices**, then the value of iteration interval indicates the amount of time to wait between executing the specified script at each slice of the currently defined Z-series. The optional argument, iteration variable, is a string that will contain the current iteration number. To use this variable, both of the optional arguments, iterations and iteration interval must be specified.
This iteration variable may then be accessed by specific commands such as ZSeriesLoad.
Back to Commands

{**-Exit**|**-x**}
This command is ignored unless sent from outside Prairie View. Closes the remote connection to Prairie View and stops processing any

script commands that follow. When sending script commands from the command line this is handled automatically, and when using PrairieLink API this is handled by calling the Disconnect method, so this command is only send explicitly when directly communicating with Prairie View via a TCP/IP connection.
Back to Commands

{**-Help**|**-?**}
Displays this document.
Back to Commands

{**-MessageToOperator**|**-mto**} <Message>
Displays the designated message to the operator in a dialog box. The script will 'hold' at this point waiting for the operator to hit the 'OK' button to continue the script execution or to hit the 'Cancel' button to abort the script.
Back to Commands

{**-NoWait**|**-nw**}
If this command is present anywhere in a string of commands to be executed, Prairie View will not wait for the commands to be processed before either: returning control to the controlling application if the commands were called from the command line or PrairieLink, or returning a DONE response immediately if the commands were sent directly over TCP/IP.
Back to Commands

{**-Shutdown**|**-xsd**}
When called any acquisitions or tasks in progress are aborted, if a remote SLM is configured the computer controlling it is shut down, and Prairie View shuts down without any confirmation prompt.
Back to Commands

{**-Silent**|**-s**}
If this command is present anywhere in a string of commands to be executed Prairie View will not pop-up any error message boxes caused by invalid commands/parameters.
Back to Commands

{**-UtilityButton**|**-ub**} <Button Index *where 1 is the first button*> {*state* **True**|**False**}
Sets the 'Utility Button' (found on the 'Misc' tab) specified by button index to the specified state.
Back to Commands

{**-Wait**|**-wt**} <Time *ms*>
Pause/wait for the specified number of milliseconds.
Back to Commands

{**-WaitForInputTrigger**|**-wfit**}
Executing this script command will cause the script to pause until an trigger signal (rising edge) is captured on the trigger hardware for Prairie View. Be careful in the use of this command. When this command is executed, the program will hang indefinitely until a trigger signal is received. While waiting for the input trigger, the script will not respond to attemps to 'abort' the script. Without the receipt of a trigger signal, the only way to abort the script would be to terminate Prairie View.
Back to Commands

File Commands

{**-LoadEnvironment**|**-le**} <Filename **.env including path*>
Load the specified environment file.
Back to Commands

{**-LoadTemplate**|**-lt**} <Filename **.env including path*>
Load the specified template file.
Back to Commands

{**-LoadImages**|**-li**} <Filename **.xml including path*>
Load the specified xml file and the associated image file(s) in playback mode.
Back to Commands

{**-LoadMarkPoints**|**-lmp**} <Series Filename **.xml including path*>

Load the Mark Point Series saved in the xml file provided as the current series, or
{**-LoadMarkPoints**|**-lmp**} <Point List Filename **.gpl including path*> [{*clear existing points* **True**|**False**}]
Load the Mark Point point locations saved in the gpl file provided, and optionally append those to the existing points instead of replacing them. By default any existing points will be cleared.
Back to Commands

{**-LoadROIFile**|**-lrf**} <Filename **.roi including path*>
Load the file containing saved ROI definitions.
Back to Commands

{**-LoadVoltageOutput**|**-lvo**} <Filename **.xml including path*>
Load the Voltage Output experiment saved in the xml file provided as the current experiment.
Back to Commands

{**-SaveEnvironment**|**-se**} <Filename **.env including path*>
Save the current environment to the file specified.
Back to Commands

{**-SetSavePath**|**-p**} <Path> [**addDateTime**]
Changes the directory where scan data is saved.

If there are any spaces in the path then it must be enclosed in "double quotations". If the optional parameter **addDateTime** is included, then the current date and time will be appended to the specified path.
Back to Commands

{**-SetFileIteration**|**-fi**} {*acquisition type* **AtlasVolume**|**BrightnessOverTime**|**LineScan**|**MarkPoints**|**PointScan**|**SingleImage**|**TSeries**|**VoltageRecording**|**WSeries**|**XZYZ**|**ZSeries**} <Iteration *an integer*>
Sets the file iteration value for the specified acquisition type.
Back to Commands

{**-SetFileName**|**-fn**} {*acquisition type* **AtlasVolume**|**BrightnessOverTime**|**LineScan**|**MarkPoints**|**PointScan**|**SingleImage**|**TSeries**|**VoltageRecording**|**WSeries**|**XZYZ**|**ZSeries**} <Filename *no path*> [**addDateTime**]
Sets the filename for the specified acquisition type. If the optional parameter **addDateTime** is included, then the current date and time will be appended to the specified filename.
Back to Commands

Action Commands

{**-ClearBOTs**|**-cb**}
Removes all saved regions of interest within the BOT option.
Back to Commands

{**-ClearROIs**|**-cr**}
Removes all saved regions of interest.
Back to Commands

{**-GetBOTRegions**|**-gb**} <Action Name>
Runs the action with the specified action name as already defined in Prairie View under “Tools/Actions…”. This command makes it possible to use a 3rd party tool to analyze an image and generate regions to be loaded into Prairie View so that their intensity can be tracked over time.
Back to Commands

{**-GetROIs**|**-gr**} <Action Name>
Runs the action with the specified action name as already defined in Prairie View under “Tools/Actions…”. This command makes it possible to use a 3rd party tool to analyze an image and generate regions of interest to be loaded into Prairie View.
Back to Commands

{**-SetActionAfterFrame**|**-af**} [<Action Name> [<Filename *including path*> [<Argument>]...]]

Sets up an action to run for the next and any subsequent frames acquired. If just an action name is given then the existing action with that name, if any, will be run. If the action name is not in use then a new one will be created using the filename and any arguments given. If the action name already exists then the action will be updated with the given filename and any arguments given. If no action name is passed then no action will be performed and any action currently being performed after frames will cease.
Back to Commands

{**-SetActionAfterScan**|**-as**} [<Action Name> [<Filename *including path*> [<Argument>]...]]
Sets up an action to run for the next and any subsequent scans. If just an action name is given then the existing action with that name if any will be run. If the action name is not in use then a new one will be created using the filename and any arguments given. If the action name already exists then the action will be updated with the given filename and and arguments given. If no action name is passed then no action will be performed and any action currently being performed after scans will cease.
Back to Commands

Image Commands

{**-ImageWindowFit**|**-iwf**}
Scales the image window(s) up or down to fit within a 512x512 pixel area on the display (this is the default image window size).
Back to Commands

{**-ImageWindowLarger**|**-iwl**} [<Percent Change *0.0-1.0*>]
Scales the image window up by the percent specified, if no percent is specified the image window(s) are scaled up 10%. Percents are based on the size of the original image in decimal format, for example '10%' would be passed in as '.1'.
Back to Commands

{**-ImageWindowOriginal**|**-iwo**}
Resizes the image window(s) to a 1:1 scale based on the current imaging resolution.
Back to Commands

{**-ImageWindowSmaller**|**-iws**} [<Percent Change *0.0-1.0*>]
Scales the image window down by the percent specified, if no percent is specified the image window(s) are scaled down 10%. Percents are based on the size of the original image in decimal format, for example '10%' would be passed in as '.1'.
Back to Commands

Visual Script Commands

{**-VisualScriptForm**|**-vsf**} <Form Title>
Creates a new window with the title provided; additional visual script commands can be used to add controls to the form. If a form has already been defined, but hasn't been shown yet, this command will first call the -VisualScriptShow (-vss) command.
Back to Commands

{**-VisualScriptLabel**|**-vsl**} <Label Text>
Adds a label with the text provided to the last form created using the -VisualScriptForm (-vsf) command. If no form had been defined, or the form has already been shown, this command does nothing.
Back to Commands

{**-VisualScriptControl**|**-vsc**} <Key> [<Index *required for indexed keys, omit otherwise*> [<Subindex *required for subindexed keys, omit otherwise*>]] [<Width>]
DO NOT USE THIS COMMAND WITHOUT EXPLICIT INSTRUCTION FROM A BRUKER FLUORESENCE MICROSCOPY REPRESENTATIVE, AS IT COULD CRASH THE SOFTWARE AND/OR POTENTIALLY DAMAGE HARDWARE. Adds a control to the last form created using the -VisualScriptForm (-vsf) command to display and edit the value of a number of parameters (see the <PVStateShard> section at the end of an Environment file for valid keys and indices). In addition to the key there are also two additional conditionally required parameters: index if the key is indexed or subindexed and subindex if the key is subindexed. For parameters with a finite number of options a dropdown control will be created and automatically sized to fit the longest option, otherwise a text box will be created with a default width of 50 pixels; an optional width parameter can be used to specify a size. If no form had been defined, or the form has already been shown, this command does nothing.
Back to Commands

{**-VisualScriptButton**|**-vsb**} <Button Text> <Script Category> <Script Name>
Adds a button with the text provided to the last form created using the -VisualScriptForm (-vsf) command. When pressed the button

will executes the specified script name within the specified script category. If no form had been defined, or the form has already been shown, this command does nothing.
Back to Commands

{**-VisualScriptNewLine**|**-vsn**}
Skips down to begin adding controls underneath those already added to the last form created using the -VisualScriptForm (-vsf) command. This can also be used multiple times to add a larger gap between controls. If no form had been defined, or the form has already been shown, this command does nothing.
Back to Commands

{**-VisualScriptShow**|**-vss**}
Finializes the the last form created using the -VisualScriptForm (-vsf) command so that no additional controls can be added, and displays the form. If no form had been defined, or the form has already been shown, this command does nothing.
Back to Commands

Acquisition Commands

{**-Abort**|**-stop**}
Aborts any scans or script commands in progress. Any script commands sent along with this command will not be run, even commands appearing before the abort; just run them in a separate request if needed. This command will also abort any acquisitions in progress, even those not started with a script. Additionally any voltage output, voltage recording or mark point experiments in progress will also be aborted. This command will have no effect when run from within Prairie View; it is only for use from the command line, through PrairieLink or direct TCP/IP communication.
Back to Commands

{**-DroppedData**|**-dd**}
This command responds with 'True' if data has been dropped during the current acquisition, otherwise it responds 'False'. This command is only useful when using either the command prompt or a TCP/IP connection since it only responds with a value and does not actually do anything.
Back to Commands

{**-GetImage**|**-gi**} <Channel *where 1 is the first image window channel*> <Process ID> <Memory Address>
This is a low level command to get live image data from Prairie View, basically the contents of the image windows, for use in third party applications. It is primarily used by PrairieLink to provide a two dimensional array of data representing the image. In order to use the command you need to know what channel of data you are interested in getting, the ID of the process requesting the data (i.e. what is displayed in the task manager), and the address of some memory that has already been allocated to hold the data. The values placed into the memory block will be 32-bit unsigned integers. When Collecting PLIM data in FLIM mode channels 5-8 contain the phosphorescence photon counts for channels 1-4 respectively.
Back to Commands

{**-LimitGSDMABufferSize**|**-lbs**} [<*enable* **True**|**False**> [<Minimum Buffer Time *ms*>]]
This command will limit the size of the DMA buffers for General Standards acquisition cards (used on systems with a resonant galvo) to a single frame, or at least 100ms (by default) if a frame period is shorter, in order to get display updates more often, or get raw data with lower latency with the ReadRawDataStream command. This is automatically set to true when using start/stop triggers to minimize data left in a buffer unprocessed when a stop trigger is received. When planning to abort an acquisition in the middle of a long sequence of frames, setting this to true could also help to ensure all the data that had been acquired thus far is saved. Shrinking the DMA buffers will increase the chance that data will be dropped since it eliminates the safety net of a larger buffer; some testing should be done to insure a specific experiment will run successfully prior to investing in an actual sample; the dropped data flag will indicate if the experiment succeeded or not. Decreasing the minimum buffer time from the default of 100ms will further increase the chance of dropping data.
Back to Commands

{**-LiveScan**|**-lv**} [{ *state* **on**|**off**}]
Begins live scanning as soon the current acquisition ends, or immediately if no acquisition is currently running. If *state* is **off** and a live scan is currently running, the live scan will be terminated. If *state* is **on** when already live scanning or if *state* is specified to be **off** when not live scanning, this script is ignored; *state* is assumed **on** if not provided. Other script commands will NOT automatically terminate the live scan as in previous versions of Prairie View.
Back to Commands

{**-MarkAllPoints**|**-slm**} [<Point Count> [{*3D* **True**|**False**}] [<X Position *% of image 0-1, where 0 is the left side*> <Y Position *% of image 0-*

1, where 0 is the top> [Z Offset *in microns, only if 3D is True*]]... *repeat Point Count times* <Laser Pulse Duration *ms*> <Laser Name *as it appears in the UI*> <Laser Power *same range as UI controls, supports 2P laser calibration*> [{*Is Spiral* **True**|**False**} <Spiral Size *% of image 0-1 in the x dimension, spiral will be forced to be a circle*> <Spiral Revolutions>] [{*Trigger* **None**|**PFI0**|**PFI1**|**PFI8**|**TrigIn**} [<Trigger Count *only used when using PFI8 trigger*>]] <Delay *ms, omit for last repetition of parameters*>]... *repeat all parameters again to mark another set of points, or specify a mask image*
Marks the specified points simultaneously on the fly using the SLM with the laser pulse parameters provided; no need to set up point locations or an experiment ahead of time. 3D and is spiral are assumed to be **False** if omitted, and if no trigger is specified it is assumed to be **None**. See Mark Points documentation for more information on specific parameters. Point count and point positions can be replaced with a mask filename and center position, see below.
{**-MarkAllPoints**|**-slm**} [<Mask Filename *including path*> <X Center *% of image 0-1, where 0 is the left side*> <Y Center *% of image 0-1, where 0 is the top*> <Laser Pulse Duration *ms*> <Laser Name *as it appears in the UI*> <Laser Power *same range as UI controls, supports 2P laser calibration*> [{*Is Spiral* **True**|**False**} <Spiral Size *% of image 0-1 in the x dimension, spiral will be forced to be a circle*> <Spiral Revolutions>] [{*Trigger* **None**|**PFI0**|**PFI1**|**PFI8**|**TrigIn**} [<Trigger Count *only used when Trigger is passed as PFI8*>]] <Delay *ms, omit for last repetition of parameters*>]... *repeat all parameters again for another mask image, or to mark another* set of points
Applies an SLM mask on the fly using the laser pulse parameters provided. The SLM mask file must be an 8-bit grayscale bitmap. Is spiral is assumed to be **False** if omitted and spiral size and revolutions should also be omitted, and if no trigger is specified it is assumed to be **None**. See Mark Points documentation for more information on specific parameters. Mask filename and center position can be replaced with a point count and point positions, see above.

{**-MarkPoints**|**-mp**} [<Category Name> <Experiment Name>]
Runs a saved Mark Point Series (or experiment) provided the category and name of the saved series/experiment; if no optional parameters are provided then the current series/experiment will be run.
{**-MarkPoints**|**-mp**} [<X Position *% of image 0-1, where 0 is the left side*> <Y Position *% of image 0-1, where 0 is the top*> <Laser Pulse Duration *ms*> <Laser Name *as it appears in the UI*> <Laser Power *same range as UI controls, supports 2P laser calibration*> [{*Is Spiral* **True**|**False**} <Spiral Size *% of image 0-1 in the x dimension, spiral will be forced to be a circle*> <Spiral Revolutions>] [{*Trigger* **None**|**PFI0**|**PFI1**|**PFI8**|**TrigIn**} [<Trigger Count *only used when Trigger is passed as PFI8*>]] <Delay *ms, omit for last repetition of parameter*>]... *repeat all parameters again to mark another location*
Marks the specified points on the fly with the laser pulse parameters provided; no need to set up a point location or experiment ahead of time. Is spiral is assumed to be **False** if omitted and spiral size and revolutions should also be omitted, and if no trigger is specified it is assumed to be **None**. See Mark Points documentation for more information on specific parameters.
Back to Commands

{**-MarkPointsMetadata**|**-mpm**} [{*enabled* **True**|**False**}]
Used to enable or disable the saving of metadata associated with ad hoc Mark Point experiments run from script commands, this has no effect on running the current Mark Point Series, or saved series, from a script command, just the experiments which are defined entirely within a script command. If no parameter is passed the saving of metadata will be enabled. Once enabled this setting persists between sessions of Prairie View, so it only needs to be set once if the preference is to always save metadata. By default this feature is disabled. Saving metadata does take some time though so for a more immediate response this feature should be disabled.
Back to Commands

{**-PointScan**|**-ps**} [<Time *ms*>]
Performs the currently defined point scan in galvo or FLIM modes. The optional time parameter, which must be an integer greater than zero, is only applicable to galvo mode and will set the point scan duration to the time specified prior to starting the point scan, and any subsequent calls without a time specified will use that same time.
Back to Commands

{**-ReadRawDataStream**|**-rrd**} <Process ID> <Memory Address> <Buffer Size *in 16-bit samples*>
This is an extremely low level command to get raw live image data from Prairie View, the raw data straight off the acquisition card, for use in third party applications. It is primarily used by PrairieLink, but could be used directly by an advanced user. In order to use the command you need to know the ID of the process requesting the data (i.e. what is displayed in the task manager), the address of some memory that has already been allocated to hold the data, and how large that block of allocated memory is. The values placed into the memory block will be 16-bit signed integers. The command will respond with the number of samples copied into the buffer, it will not wait until the buffer is filled; if no samples were added since last called it will respond with a zero. Consecutive calls of this command will return a continuous stream of whole frames. This command will do nothing until the StreamRawData command is sent with a parameter of true; this command starts caching raw data for subsequent acquisitions.
Back to Commands

{**-SetAcquisitionMode**|**-sam**} {*mode* **AOD**|**Camera**|**FLIM**|**Galvo**|**Resonant Galvo**|**SFC**|**Spiral**}
Switches the current acquisition mode to the mode specified by the mode parameter. Depending on the transition, the user may need to place a wait command after this command to give time for the system to complete the switch
Back to Commands

{**-SingleScan**|**-ss**} [{*autosave* **True**|**False**}]
Performs a single scan. Passing the optional autosave parameter as **False** will not increment the file iterator after the scan causing the next single scan to overwrite the data.
Back to Commands

{**-SingleScanTriggered**|**-sst**} [{*autosave* **True**|**False**}]
Performs a single scan after receiving an input trigger. Passing the optional autosave parameter as **False** will not increment the file iterator after the scan causing the next single scan to overwrite the data.
Back to Commands

{**-StreamRawData**|**-srd**} [{**True**|**False**} [<Buffer Size *in frames*>]]
This is an extremely low level command to enable raw data streaming so that the raw data can be retrieved using the ReadRawDataStream command. Passing **True**, or nothing, will enable raw data streaming and passing **False** will disable raw data streaming.

The optional buffer size allocates a ring buffer of up to 65,535 frames worth of data to be acquired into; if data is not read out of the ring buffer fast enough the buffer will fill and no more data will be added, but it will still be possible to read out the remainder of the data which was written. Omitting the buffer size, or passing it as zero, will result in a buffer consisting of three frames juggling the most recently collected data with no possibility of overrunning the buffers; each new frame overwrites and older one, and the third is set aside to return the remainder of any frame which has been partially read.

A larger buffer is useful for an application that needs to see all the data and can process it quickly before the buffer fills. If the data processing begins to lag behind the acquisition a larger buffer can buy more time, or possibly contain the entirety of the acquisition, but the data would no longer be processed in real time making it impossible to react to the data.

The default three frame juggling buffer will still return all the data if polled quickly enough, but was really meant to benefit applications which only poll for data periodically and need to make a decision based on the most recent data. For example wait for some indefinite period of time after the acquisition starts, quickly analyze one or more of the most recent frames in real time, do something based on that analysis, and then wait another indefinite time period before looking at the most recent data again.
Back to Commands

{**-TSeries**|**-ts**}
Performs a T-series using the current settings.
Back to Commands

{**-TSeriesLoad**|**-tsl**} [<Filename **.env including path*>]
Loads the T-Series definition from the environment file provided. If no filename is provided the T-Series definition will be loaded from the one saved the last time Prairie View was closed.
Back to Commands

{**-WaitForScan**|**-w**}
Prairie View will wait for the current scan to complete before moving on to the next command. If there isn't a scan currently in progress this command will do nothing.
Back to Commands

Line Scan Commands

{**-GetFreehandLine**|**-gl**} <Action Name>
Runs the action with the specified action name as already defined in Prairie View under "Tools/Actions…". This command makes it possible to use a 3rd party tool to analyze an image and generate a free hand line scan pattern to be loaded into Prairie View.
Back to Commands

{**-LineScan**|**-ls**} [{*append* **True**|**False**}]
Performs a line scan acquisition. Setting the parameter *append* as **True** will append the line scan acquisition data to the last line scan acquisition acquired.
Back to Commands

{**-LineScanDialog**|**-ld**} {**open**|**close**}
Opens of closes the line scan dialog depending on whether **open** or **close** is passed as an argument.
Back to Commands

{**-LineScanLines**|**-lsl**} <Number of Lines>
Sets the number of lines to scan in the line scan dialog.
Back to Commands

{**-LineScanMode**|**-lm**} {*Line Type* **Circle**|**Freehand**|**Line**|**Lissajous**|**Segmented**|**Spiral**}
Set the desired line type for a line scan acquisition.
Back to Commands

{**-LoadLineScan**|**-lls**} <Filename **.xml including path*>
Loads the line scan definition saved in the specified file.
Back to Commands

Z-Series Commands

{**-FindSlice**|**-fs**} <Channel *where 1 is the first image window channel*> <Slice Thickness *negative to go opposite direction*> <Intensity Threshold *% of previous slice intensity, <100 dimmer, >100 brighter*> <Stop Limit> <Backoff> {*return to start if limit reached* **True**|**False**}
Steps through a sample slice by slice starting at the current position and moving as determined by the slice thickness. The search process will end when the stop limit is reached or if a slice meets or exceeds appropriate intensity threshold before that. The intensity threshold comparison is based upon the intensity value from the first image and the intensity value from the current image. Warning: Failure to set an appropriate stop limit may cause damage to sample and/or hardware if physical limits are reached and/or exceeded. The backoff argument allows the operator to specify an amount for the z motor to move from its current location (in microns) before starting the search. The return to start if limit reached argument is **True**, will have the z motor move back to the position it was at when the search was started (ignoring the value of backoff), if the stop limit is reached. If the return to start if limit reached argument is **False**, then if the stop limit is reached, the z motor will remain at that location.
Back to Commands

{**-SetZSeriesDisplay**|**-zsd**} [{*Display Option* **All**|**Middle**|**Specific**} [<Slice Number *only used for specific display option*>]]
Sets the display option for the current Z-series definition to show **All** slices, only the **Middle** slice, or a **Specific** slice number. If the option is **Specific** then the second parameter should be the slice number to display (from 1 to the number of slices). If no parameters are passed the **All** display option will be used.
Back to Commands

{**-SetZSeriesNumberOfSlices**|**-zsn**} <Number of Slices>
(This command is only valid if the current Z-Series step mode is fixed)
Sets the number of slices in the current Z-series definition.
Back to Commands

{**-SetZSeriesStart**|**-zsb**} [{**onlyMotors**|**allSettings**} [{*Reset Z-Series Stop* **True**|**False**}]]
(This command is only valid if the current Z-Series step mode is fixed)
Sets the starting position for the current Z-series definition. If the optional parameter, **onlyMotors** is used, then only the motor positions related to the Z-series definition will be updated for the start position. If this parameter is missing, or is set to **allSettings**, then all of the start position parameters will be defined based upon the current motor positions and control settings (laser and PMT settings). If the second optional parameter, reset Z-Series stop, is **True** then the stop position of the Z-series definition will be adjusted by the amount that the start position was adjusted. If this value is **False**, or omitted, then the stop position of the Z-series will not be adjusted from its current value.
Back to Commands

{**-SetZSeriesStepSize**|**-zsz**} <Step Size>
(This command is only valid if the current Z-Series step mode is fixed)
Sets the step size for the current Z-series definition.
Back to Commands

{**-SetZSeriesStop**|**-zse**} [{**onlyMotors**|**allSettings**} [{*Reset Z-Series Start* **True**|**False**}]]
(This command is only valid if the current Z-Series step mode is fixed)
Sets the ending position for the current Z-series definition. If the optional parameter, **onlyMotors** is used, then only the motor positions related to the Z-series definition will be updated for the stop position. If this parameter is missing, or is set to **allSettings**, then all of the stop position parameters will be defined based upon the current motor positions and control settings (laser and PMT settings). If the second optional parameter, reset Z-Series start, has a value of **True** then the start position of the Z-series definition will be adjusted by the amount that the stop position was adjusted. If this value is **False**, or ommitted, then the start position of the Z-series will not be adjusted from its current value.

Back to Commands

{**-ZSeries**|**-zs**}
Performs a Z-series using the current settings.
Back to Commands

{**-ZSeriesAddSlice**|**-zsas**}
(This command is only valid if the current Z-Series step mode is variable)
Add this position as a slice at the end of the list for a variable step mode Z-series.
Back to Commands

{**-ZSeriesClear**|**-zscl**}
Removes all slices/positions/compensation values from the current Z-Series.
Back to Commands

{**-ZSeriesInsertSlice**|**-zsis**} <Index>
(This command is only valid if the current Z-Series step mode is variable)
Insert a new slice, using the current position, in the list for a variable step mode Z-series at the provided index (index must be between 0 and the number of slices currently in the list)
Back to Commands

{**-ZSeriesLoad**|**-zsl**} <Name>
Load the saved Z-series definition with the name given as the current Z-series. Alternatively, when combined with the ExecuteScript command, the name parameter can be the iteration variable. If the iteration variable is used, then the Z-series loaded will be based upon the order of the current list of Z-series definitions.
Back to Commands

{**-ZSeriesMoveTo**|**-zsmt**} {**Start**|**Middle**|**Stop**|<Slice Number>}
Move the focus assembly to the start, middle, or stop position or a specified slice number as defined by the current Z-series. In addition to moving the focus assembly to the specified location, if laser power and/or PMT gain compensation is enabled for the current Z-series, then the laser and PMT settings will be set to the values corresponding to the selected location.
Back to Commands

{**-ZSeriesRemoveSlice**|**-zsrs**} <Index>
(This command is only valid if the current Z-Series step mode is variable)
Removes the slice at the specified index (between 0 and one less than the number of slices) from the list for a variable step mode Z-series
Back to Commands

{**-ZSeriesSave**|**-zss**} <Name>
Saves the current Z-series definition with the name given.
Back to Commands

{**-ZSeriesStepMode**|**-zssm**} <*step mode* **Fixed**|**Variable**>
Sets the current Z-Series step mode to the value provided in the *step mode* parameter. If Atlas Imaging is active this command will have no effect (only fixed step mode Z-Series is available while Atlas Imaging is active).
Back to Commands

Scan Setting Commands

{**-EnterROI**|**-er**} <X Location *% of FOV 0-1, where 0 is the left side*> <Y Location *% of FOV 0-1, where 0 is the top*> <Width *% of FOV 0-1*> <Height *% of FOV 0-1*>
Enters a region of interest to be scanned based on the location and dimensions provided based on the source image (scan area when not scanning a region of interest).
Back to Commands

{**-GetState**|**-gts**} <Key> [<Index *required for indexed keys, omit otherwise*> [<Subindex *required for subindexed keys, omit otherwise*>]]
Returns the value of a number of parameters (see the <PVStateShard> element at the end of an Environment file for valid keys, indices and subindices). In addition to the key there are also two additional required parameters: index if the key is indexed or subindexed and subindex if the key is subindexed. This command is only useful when using either the command prompt or a TCP/IP connection since it

only responds with a value and does not actually do anything.
Back to Commands

{**-ParameterSet**|**-pa**} <Name> [<Track>]
Applies settings from the Parameter Set which matches the name provided. For multi-track Parameter Sets an optional track number can be specified which defaults to the first track.
Back to Commands

{**ROILoad**|**-roi**} {<Name>|**noROI**}
Load the ROI definition with the specified name, or if **noROI** is passed exit the current ROI if applicable. Alternatively, when combined with the ExecuteScript command, the name parameter can be the iteration variable. If the iteration variable is used, then the ROI loaded will be based upon the order of the current list of ROI definitions.
Back to Commands

{**-SamplesPerPixel**|**-spp**}
Gets the current number of samples acquired for each pixel in the image. This command is useful in conjunction with the ReadRawDataStream command to figure out how to parse the raw data stream. This command is only useful when using either the command prompt or a TCP/IP connection since it only responds with a value and does not actually do anything.
Back to Commands

{**-SetChannel**|**-c**} [<Channel *where 1 is the first image window channel*> {**On**|**Off**}]...
Enables and disables channels on the first image window. Multiple channels can be be turned on or off with the same command by repeating parameters.
Back to Commands

{**-SetDwellTime**|**-dt**} <Dwell Time>
Sets the dwell time as close as possible to the given dwell time value. Minimum dwell time and sampling rate limit potential dwell time values. A warning will be displayed if the specified dwell time cannot be achieved.
Back to Commands

{**-SetFrameAveraging**|**-fa**} <Frames to Average>
Sets the frame averaging to the specified value.
Back to Commands

{**-SetImageSize**|**-is**} <Width> [<Height>]
Sets the image size as close as possible to the given dimensions. Minimum pixels per line limits potential image widths. A warning will be displayed if the specified image size cannot be achieved. If height is not specified then the width will be used for both dimensions.
Back to Commands

{**-SetOpticalZoom**|**-oz**} <Optical Zoom>
Sets the optical zoom to the specified value.
Back to Commands

{**-SetScanRotation**|**-sr**} <Angle>
Sets the scan rotation to the specified angle.
Back to Commands

{**-SetState (-sts)** <Key> <Value> [<Index *required for indexed keys, omit otherwise*> [<Subindex *required for subindexed keys, omit otherwise*>]] [{*force modified* **True**|**False**}]
DO NOT USE THIS COMMAND WITHOUT EXPLICIT INSTRUCTION FROM A BRUKER FLUORESENCE MICROSCOPY REPRESENTATIVE, AS IT COULD CRASH THE SOFTWARE AND/OR POTENTIALLY DAMAGE HARDWARE. Changes the value of a number of parameters (see the <PVStateShard> element at the end of an Environment file for valid keys, indices and subindices). The two required parameters are a key and a value, there are also two additional required parameters: index if the key is indexed or subindexed and subindex if the key is subindexed. The force modified parameter specifies whether or not to mark the value as being changed even if the value is the same, this should always be omitted unless there is a specific reason for passing it.
Back to Commands

{**-SetCustomOutput**|**-co**} <Index *where 0 is the first defined custom output*> [<Percent> <Value>]...
Defines the custom output to be used with upcoming acquisitions. The index parameter is used to specify which custom output definition to replace. The percent parameter specifies the percent of a scan line during which to output the value specified. Additional percent and value pairs can be repeated to define more complex waveforms.

Back to Commands

Hardware Commands

{**-Camera**|**-ca**} <Feature> <Value> [<Index *where 0 is the first camera*>]
Change the value of the specified feature, see table below. The particular values are going to be dependent upon the actual camera. The quickest way to know the proper value or range for a given feature for a camera, would be to look at the corresponding control within Prairie View. If there are multiple cameras an index can be used to specify which one for certain features.

Feature	Value	Index
{**ExposureTime**|**et**}	<1..n>	
{**ExposureMode**|**em**}	{**Bulb**|**Timed**|**TriggerAll**|**TriggerFirst**}	
{**BinFactor**|**bf**}	{**1**|**2**|**4**|**8**}	
{**Gain**|**gn**}	<0..100>	{**0**|**1**}
{**GainMultFactor**|**gnmf**}	<0..4095>	{**0**|**1**}
{**ReadOutPort**|**rop**}	{**0**|**1**}	{**0**|**1**}
{**CameraFan**|**fan**}	{**On**|**Off**}	{**0**|**1**}

Back to Commands

{**-CenterGalvos**|**-cg**} [{**True**|**False**}]
This command will center the imaging galvos. Passing optional parameter as **False** will return the galvos to their parked positions, otherwise the galvos will be centered.
Back to Commands

{**-GetETLValue**|**-ge**} <Position *in microns*>
This command will return the drive value for the ETL for a specific position in microns. This is to be used in conjunction with the SetCustomOutput command to synchronize the position of the ETL with an acquisition. This command is only useful when using either the command prompt or a TCP/IP connection since it only responds with a value and does not actually do anything.
Back to Commands

{**-GetMotorPosition**|**-gmp**} <*axis* **X**|**Y**|**Z**> [<Index *where 0 is the first device*>]
Returns the current position (as displayed on the main form) for a motor driving the specified axis. If there are multiple motors/devices used for a particular axis the optional index parameter (zero indexed) can be used to specify a specific device, the default index will be that of the currently active device. This command is only useful when using either the command prompt or a TCP/IP connection since it only responds with a value and does not actually do anything.
Back to Commands

{**-MoveMotor**|**-mr**} [<*axis* **X**|**Y**|**Z**> <Distance to Move *in microns*> [<Index *where 0 is the first device*>]]... *repeat for multiple axes* [{*Wait* **True**|**False**}]
Moves a motor for a specified axis by a specified distance. If there are multiple motors/devices used for a particular axis the optional index parameter (zero indexed) can be used to specify a specific device, the default index will be that of the currently active device. This can be repeated for multiple axes. The last optional Wait parameter is used to specify whether to wait until all motors specified have moved to their target location, the default value is **False**.
Back to Commands

{**-NikonNiMicroscope**|**-nni**} <Device> <Value>
Change the value of the specified device, see table below. For devices that have a value range of 1..n, n would be replaced by the number of discrete positions for the device. For example, if there are 6 motorized nosepiece positions, the valid options for value are 1, 2, 3, 4, 5, and 6.

Device	Value
{**FilterCassetteBlock1**|**fbc1**}	<1..n>
{**FilterCassetteBlock2**|**fbc2**}	<1..n>
{**CondenserCassette**|**cc**}	<1..n>
{**LightPathDriver**|**lpd**}	<1..n>
{**TLHalogenLampSoftwareControl**|**thhlsc**}	{**True**|**False**}

{**TLHalogenLampState**|**thhls**}	{**True**|**False**}
{**TLHalogenLampPower**|**thhlp**}	<0..24>
{**MotorizedNosepiece**|**mnp**}	<1..n>

Back to Commands

{**-NikonTiMicroscope**\|**-nti**} <Device> [<Value *optional based upon device*>]
Change the value of the specified device, see table below. For devices that have a value range of 1..n, n would be replaced by the number of discrete positions for the device. For example, if there are 6 motorized nosepiece positions, the valid options for value are 1, 2, 3, 4, 5, and 6.

Device	Value
{**FilterCassetteBlock1**|**fbc1**}	<1..n>
{**FilterCassetteBlock2**|**fbc2**}	<1..n>
{**CondenserCassette**|**cc**}	<1..n>
{**LightPathDriver**|**lpd**}	<1..n>
{**TLHalogenLampSoftwareControl**|**thhlsc**}	{**True**|**False**}
{**TLHalogenLampState**|**thhls**}	{**True**|**False**}
{**TLHalogenLampPower**|**thhlp**}	<0..24>
{**MotorizedNosepiece**|**mnp**}	<1..n>
{**PerfectFocusState**|**pfs**}	{**True**|**False**}
{**PerfectFocusOffset**|**pfo**}	<0..40000>
{**PerfectFocusMemory**|**pfm**}	
{**PerfectFocusRecall**|**pfr**}	

Back to Commands

{**-PanGalvo**\|**-png**} {*Axis* **X**\|**Y**\|**Center***)*} {*Distance to Pan* **Coarse**\|**Medium**\|**Fine**\|<Microns *if objective is calibrated, otherwise Volts*>}
Pans the imaging galvos the specified distance in either **X** or **Y.** If the scan is rotated the panning will occur in both axes such that the image window pans the expected direction. Pan values can be either **Coarse** (0.25 V), **Medium** (0.1 V), **Fine** (0.01 V) or any numeric distance for finer or more specific control. Numeric distances will be interpreted as microns when the selected objective lens has been calibrated, otherwise distances will be interpreted as Volts. A negative value will pan up or left and a positive value will pan down or right, for example **-png x -coarse** will pan left by a large 0.25V step. Passing **Center** for the axis will reset the pan offsets for both axes back to the center.
Back to Commands

{**-SendGPIOCommand**\|**-gc**} <Command> [<Argument>]...
Sends the specified command directly to the GPIO controller. Contact a Bruker Fluorescence Microscopy representative for more details.
Back to Commands

{**-SendMAMCCommand**\|**-mc**} <Command> [<Argument>]...
Sends the specified command directly to the multi-axis motor controller. Contact a Bruker Fluorescence Microscopy representative for more details.
Back to Commands

{**-SendPiezoCommand**\|**-pc**} <Command> [<Argument>]...
Sends the specified command directly to the prairie piezo controller. Contact a Bruker Fluorescence Microscopy representative for more details.
Back to Commands

{**-SendServoCommand**\|**-sc**} <Command> [<Argument>]...
Sends the specified command directly to the quad servo controller. Contact a Bruker Fluorescence Microscopy representative for more details.
Back to Commands

{**-SendResonantCommand**\|**-rc**} <Command> [<Argument>]...
Sends the specified command directly to the resonant galvo controller. Contact a Bruker Fluorescence Microscopy representative for

more details.
Back to Commands

{-**SetMotorPosition|-ma}** [<*axis* **X|Y|Z**> <Position> [<Index *where 0 is the first device*>]]... *repeat for multiple axes* [{*Wait* **True|False**}]
Moves a motor for a specified axis to a specified position. If there are multiple motors/devices used for a particular axis the optional index parameter (zero indexed) can be used to specify a specific device, the default index will be that of the currently active device. This can be repeated for multiple axes. The last optional Wait parameter is used to specify whether to wait until all motors specified have moved to their target location, the default value is **False**.
Back to Commands

{**-SetObjectiveLens**|**-sol**} <Name>
Sets the objective lens selection to the objective with the name specified.
Back to Commands

{**-SFC**|**-sfc**} <Feature> <Value>
Change the value of the specified feature, see table below. The particular values are going to be dependent upon the actual camera. The quickest way to know the proper value or range for a given feature for the SFC, would be to look at the corresponding control within Prairie View. If there are multiple cameras an index can be used to specify which one for certain features.

Feature	Value	Index
{**ExposureTime**|**et**}	<1..n>	
{**ExposureMode**|**em**}	{**Bulb**|**Timed**|**TriggerAll**|**TriggerFirst**}	
{**BinFactor**|**bf**}	{**1**|**2**|**4**|**8**}	
{**Gain**|**gn**}	<0..100>	{**0**|**1**}
{**GainMultFactor**|**gnmf**}	<0..4095>	{**0**|**1**}
{**ReadOutPort**|**rop**}	{**0**|**1**}	{**0**|**1**}
{**EmissionFilter**|**ef**}	<0..n>	
{**Aperture**|**ap**}	<0..n>	
{**CameraFan**|**fan**}	{**On**|**Off**}	{**0**|**1**}

Back to Commands

{**-SetCalibrationMask**|**-scm**} <Region Number *0..63*> <Stripe Depth *0..255*> [<Region Count>]
Applies a calibration mask to the SLM. Region count must be the square of an integer, for example 8^2 = 64 which is also the default if the parameter is omitted.
Back to Commands

{**-ZDeviceControl**|**-zdc**} [{**External**|**Software**}]
Sets the current Z device for either **Software** control via Prairie View or for **External** control. **External** control means that the Z device will be in a state to be controlled via an analog voltage level. Not all Z devices support this feature. Passing no parameters will assume **Software** control.
Back to Commands

{**-ZeissMicroscope (-zm)** <Device> <Value>
Change the value of the specified device, see table below. For devices that have a value range of 0..n, n would be replaced by one less the number of discrete positions for the device. For example, if there are 6 motorized nosepiece positions, the valid options for value are 0, 1, 2, 3, 4, and 5.

Device	Value
{**ReflectorChanger**|**rc**}	<0..n>
{**TLShutter**|**tls**}	{**True**|**False**}
{**TLHalogenLampState**|**tlhls**}	{**True**|**False**}
{**TLHalogenLampPower**|**tlhlp**}	<0..100>
{**RLShutter**|**rls**}	{**True**|**False**}
{**RLHalogenLampState**|**rlhls**}	{**True**|**False**}
{**RLHalogenLampPower**|**rlhlp**}	<0..100>
{**SideportTurret**|**spt**}	<0..n>

{**Optovar**|**op**}	<0..n>
{**MotorizedNosepiece**|**mnp**}	<0..n>

Back to Commands

Laser/PMT Commands

{**-AlternateBeamRoute**|**-abr**} <Index *where 0 is the first alternate beam route*> {*enabled* **True**|**False**}
Enables or disables the alternate beam route specified by the index. The alternate beam route controls are found on the Power/Gain tab under the laser power sliders.
Back to Commands

{**-SecondaryLaserBeamRoute**|**-slbr**} {*enabled* **True**|**False**}
Legacy command replaced by AlternateBeamRoute. Enables or disables the first defined alternate beam route. The alternate beam route controls are found on the Power/Gain tab under the laser power sliders.
Back to Commands

{**-SetLaserPower**|**-lp**} {<Name>|<Index *where 1 is the first defined laser*>} {<Power>|**By**} [<Power Offset *only applies when By is passed as second parameter*>]
Changes the power for the laser specified by either it's name or index. To change the laser power to a specific value that power will be the second parameter, otherwise to make a relative change in power pass **By** as the second parameter and the relative power offset as the optional third parameter. The power offset can be either positive to increase power or negative to decrease power.
Back to Commands

{**-SetMultiPhotonWavelength**|**-mpw**} <Wavelength *nm*> [<Index *where 1 is the first 2P laser defined*>]
Sets the operating wavelength for the multi-photon laser specified by the index. If no index is passed the first defined multi-photon laser will be assumed.
Back to Commands

{**-SetPMTGain**|**-pg**} <Index *where 1 is the first defined PMT*> {<Gain>|**Zero**|**Previous**|**By**} [<Gain Offset *only applies when By is passed as second parameter*>]
Sets the gain for a specified PMT. For the second parameter: a numerical value will just set the gain to the value specified, **Zero** will set the gain to 0 and store the previous value to be recalled with **Previous**, **By** will indicate that a third optional parameter will be provided specifying a relative gain offset. The gain offset can be either positive to increase gain or negative to decrease gain.
Back to Commands

Shutter Commands

{**-OverrideHardShutter**|**-ohs**} {<Name>|<Index *where 0 is the first shutter defined*">} {*state* **Open**|**Close**|**Auto**}
Forces a specific shutter, specified by a zero indexed number or by name, to remain either **Open** or **Close**d until instructed otherwise by either: this command, or from the Maintenance dialog. Setting the state to **Auto** will resume normal operation. This command can be used to keep a shutter open for the duration of a scripted experiment, otherwise the default behavior is to open and close the shutter for each separate acquisition.
Back to Commands

{**-SetHardShutter**|**-hrd**} {*state* **Open**|**Close**}
Opens or closes the imaging hard shutter.
Back to Commands

{**-SetSoftShutter**|**-sft**} {*state* **Open**|**Close**}
Opens or closes the imaging soft shutter.
Back to Commands

Stage Position Commands

{**-AddStagePosition**|**-spa**} [<X Position> <Y Position> <Z Position>]...
Adds one or more stage locations passed in sets of 3 coordinates X, Y and Z. In case of multiple Z devices the active Z device denoted in the stage control section of the main form will be used, other Z devices will retain their current positions. If no parameters are passed

then the current stage location will be added.
Back to Commands

{**-ClearStagePositions**|**-spc**}
Clears all existing saved stage locations.
Back to Commands

{**-LoadStagePositionFile**|**-lspf**} <Filename **.xy including path*>
Load the file containing saved XYZ stage positions.
Back to Commands

{**-MoveToStagePosition**|**-mtsp**} <Index *where 1 is the first defined XY stage position*>
Moves the stage to the specified position in the list of stage positions found on the XY Stage tab. Alternatively, when combined with the ExecuteScript command, the index parameter can be the iteration variable. If the iteration variable is used, then the stage position moved to will be based upon the order of the current list of saved stage positions.
Back to Commands

{**-SaveStagePositionFile**|**-sspf**} <Filename **.xy including path*>
Save a file containing saved XYZ stage positions.
Back to Commands

{**-SetGridLocations**|**-sgl**}
Generates a grid of XY Stage locations to cover all the currently defined stage locations, and the area between them, and replaces those locations with a regularly spaced grid. The number of locations in that grid and their spacing depends on the current grid overlap which can be changed with the SetGridOverlap command. This is equivalent to pressing the Generate Montage button in Atlas Imaging.
Back to Commands

{**-SetGridOverlap**|**-sgo**} <Overlap *% 0-50*>
Defines the amount of overlap, as a percentage of the field of view based upon the current objective lens selection, for Atlas Imaging.
Back to Commands