**Spec Infoserver (SIS) Documentation for v2.26 (10-2012).**

**Installation Notes:**

* In addition to the SpecServ Linux executable, SIS requires the shared memory macro shmem.mac to be installed (in SPEC), and SPEC to be run in server mode, e.g. “SPEC –S 2033” (port 2033)
* SpecServ.cfg controls important settings of SIS and should be adjusted for each beamline.

**Running Spec Infoserver:**

./specserv localhost:xxxx where xxxx is the port number SPEC is listening on, e.g. 2033.

Note that this is not the port that is used by SIS clients! The default port number that SIS listens on is 2034.

**Spec Infoserver Command Reference**

**Opcode** | Function  
=====================================================================================  
**?ver** | version information

Example: “?ver” returns: “Spec Infoserver *version information*”

**?usr** | name of the current SPEC user

**?con** | retrieve text from the SPEC's console output buffer (screen output).

**?con idx |** returns current buffer index (running)

Example: “?con idx” returns: “1280”

**?con, ?con all |** return the entire buffer (1000 lines max)

Example: “?con” returns:

Getting configuration parameters from SPECD/spec/config.  
Using Galil MC2280sG7a at "192.168.15.21".  
Using Galil MC2180 Rev 1.0g at "192.168.15.22".  
Using Galil MC2180 Rev 1.0g at "192.168.15.23".  
Using Galil MC2180 Rev 1.0i at "192.168.15.24".  
Using JORWAY 73A Rev 300.  
Using macros for a single-motor monochromator.

**?con N** | retrieve the last N entries from the buffer

Example: “?con 2” returns: “

Using JORWAY 73A Rev 300.  
Using macros for a single-motor monochromator.

**?con N-** | retrieve all entries from buffer index N (zero based)

Example: “?con 5-” returns: “

Using JORWAY 73A Rev 300.  
Using macros for a single-motor monochromator.

A trailing ‘z’ character, added to the first argument sends the results gzipped and Base64-encoded (RFC 3548). Examples: “?con allz”, “?con 10z”

**?avl** |Is SPEC available?, e.g. to receive a !cmd string, 1 if motor is available, 0 if busy. See also 17th bit of the value returned by **?sta**

Example: “?avl” returns: “1”

**?bsy** | Is SPEC busy?, inverse of **?avl**

**?mp** | motor position of single motor, argument mne

Example: “?mp tth” returns: “1.205” (Note: As in SPEC, this is the user dial)

**?mi** | single motor information, argument mne, returns 4 bit value:

2^0: motor moving (0-no|1-yes)  
2^1: motor disabled (0-no|1-yes)  
2^2: low limit active (0-no|1-yes)  
2^3: high limit active (0-no|1-yes)

Example: “?mi tth” returns: 1 (motor moving happily)

**?mne** | all motor mnemonics, comma separated

Example: “?mne” returns: “tth, th, table, h0gap, v0gap”

**?mpa** | all motor positions, comma separated

Example: “?mpa” returns: “1.205, 0.6025, -38.777, 2.0, 0.1”  
  
**?mia** | status information from all motors, comma separated, for return value see “?mi” above

Example: “?mia” returns: “1, 1, 0, 0, 0”

**?cta** | all counters (values of last counting event)

Example: “?cta” returns: “4367, 1001, 23489234, 324324”

**?det** | detector status string (for beamlines that support detector status information, e.g MAR on BL 11-3, 1-4, 1-5)

**?sta** | status flag (32-bit unsigned integer); 17th bit is high when SPEC is busy, 18th bit is set when client has control. Note, that most flags are reserved for use by the specific beam line macros (see also shmem.mac)

**?all** | motor positions & statuses, counter values & statues, comma separated

Example: “?all” returns: “1.205, 0.6025, -38.777, 2.0, 0.1, 1, 1, 0, 0, 0, 4367, 1001, 23489234, 324324, 1, 1, 1, 1”

“?all z” sends the results gzipped and Base64-encoded (RFC 3548).

**?sci** | Details on the current/most recent scan, i.e. scan type, number of data points, current data point idx, etc

**?plt** | SPEC data from current plot, comma separated datapoints

**?plt idx** | returns current buffer index (running)

Example: “?plt idx” returns: “1280”

**?plt, ?plt all** | return the entire buffer (4096 pts max)

**?plt N** | retrieve the last N rows of data points

**?plt N-** | retrieve all data points beginning from row N (zero based)

A trailing ‘z’ character, added to the first argument sends the results gzipped and Base64-encoded (RFC 3548). Examples: “?plt allz”, “?plt 10z”

**!cmd** | Sends SPEC command (SIS client needs to be in control – see below).

The command string can be any valid SPEC command with a few exceptions that are not permitted, either because they would circumvent access control or break the client connection. The forbidden commands are: “remcon”, “config”, “help”, “quit”. Returns either: “client not in control.” if the client does not have the control token, or: “#1, command sent.” The number is the ticket number that ties the command to the result that is retrieved by the **?res** command.

Example: “!cmd mv tth 10” returns: “#17, command sent.” (if it was the 17th !cmd command)

**?res** | Retrieves the result/return value of the last issued command (**!cmd**). Returns either “#N, OK, retval” where retval is the value returned by the last command, or “#N, OK, errdescr” in case the last command resulted in an error (errdescr contains SPEC error message)

Example: “?res” after “!cmd mv tth 10” returns: “#1, OK, 0”

“?res” after “!cmd 2+2” returns: “#2, OK, 4”

“?res” after “!cmd flkjfl” returns: “#2, Err, Not a command or macro: "flkjfl"”

**!abr** | Sends abort command to SPEC. Identical to Ctrl+C press on the SPEC console.

**!log** | internal log file-related

**!unx** | Sends a unix command to server machine (runs with the same permissions as SIS). Note: this is dangerous and should be avoided!  
Returned (comma-separated) are the Linux error code (0 when the command execution was successful) and the return value of the unix command.

Example: “!unx uname –a” returns: “0, Linux bl15lx.slac.stanford.edu 2.6.18-308.13.1.el5…”

**!rqc** | Requests control of the SPEC machine. Only 1 SIS client can send ! commands such as “!cmd”. Even then, control is only available if given to the client by the user at the beamline by “remcon on”. Note, that on some beamlines, “remcon on” is the default so that GUIs can run. The user of the SPEC console can revoke control at any time by “remcon off”. A subsequent “remcon on” does not automatically return control to any client previously in control, but makes it again available on a first come first serve basis.  
Returns either “control not available.” or “client in control.” when the request was successful.

Example: “!rqc” returns: “control not available.”

**!rlc** |Release control; returns “client not in control.”

Example: “!rlc” returns: “client not in control.”

**?inc** | does client have control? (see also 18th bit from **?sta**). Returns 0 if the client does not have control, 1 otherwise (client is in control).

Example: “?inc” returns: “0”