Undirected commands	
Step the simulation	step (this is for synchronous mode only. if mars is running asynchronously, you can ignore this.)
Go ahead	swb (swb = "step while busy")
What agents are available?	agents
what time is it?	declarations (note that the response string from "declarations" must be parsed to get the simulation time (simStep). we need to think about what alfred's information parsing capabilities will be.)

Note that the "undirected" commands are actually handled by the dispatch agent that handles the TCP connection with the outside world. Currently, the Mars simulation is configured to work in synchronous mode, meaning that time is not advanced in the simulation unless the dispatch agent is instructed to do so. Synchronous mode has various advantages to debugging but asynchronous mode does exist.

The issue of parsing responses from a target system is one that Alfred will presumably need to address. The "declarations" command queries a structure maintained by the simulator substrate that is basically a blackboard that any agent in the simulation can use to count stuff. The simulation timer is called "simStep". The response to "declarations" is an annotated list { … simStep=X … } and so a simple parser can answer the question "what time is it?", but that parser has to come from somewhere.

Rover Commands (*xxx* = agent name)	
acknowledge	send *xxx* ack
send sensor readings	send *xxx* sv
what is your location?	send *xxx* sv (note that the response string from "sv" must be parsed to get the desired sensor reading (in this case, "location")…)
take panoramic images	send *xxx* goal* pano() (* – could be action instead of goal)
take panoramic images at waypoint 4	send *xxx* goal pano() loc 4
take panoramic images at time 113	send *xxx* goal pano() at 113
localize	send *xxx* goal* localize()
calibrate on three	send *xxx* goal cal(3)
transmit images	send *xxx* goal* xmit()
do science at 2	send *xxx* goal science(2)
move to 6	send *xxx* goal moveto(6)
recharge	send *xxx* goal charge()

Here we are more or less assuming that the human agent intends all commands to be directed at the agent *xxx*. The difference between "action" and "goal" is that the planner will be invoked in all cases where the goal command is used. This allows the rover to move to an appropriate location for recharging, science, etc. as well as automatic calibration when science is the intended goal. One could reasonably get away with always using the goal command – unless, of course, the human user made

explicit that a certain utterance had the intent to use the “action” semantics. There are rarely, if ever, any practical advantages to using “action” over “goal”, though it is useful in inducing expectation violations when debugging. Note that the “action” command does not process the “loc” or “at” clauses, and so this command literally attempts the specified action in the current state. It will fail if the preconditions for success have not been met.

The instances above where goal is marked with an asterisk indicate the situations where “action” is always interchangeable with “goal”. Basically, when the intended action has no preconditions for success, the “action” command can be used without unfortunate consequences.