UNIVERSITÀ DI PISA

Distributed Systems and Middleware Technologies

JANET Home Simulator User Guide

Table of Contents

JANET Simulator Commands Syntax

The complete set of commands offered by the JANET Simulator is presented in the tables below, with their parameters conforming the following syntax:

- A lower-case parameter (i.e., an atom) should be left as it is:

  `jsim:print_nodes(stopped)` → `jsim:print_nodes(stopped)`

- An upper-case parameter not enclosed in quotes denotes a non-string variable:

  `jsim:print_ctr_tree(Loc_id)` → `jsim:print_ctr_tree(5)`

- An upper-case parameter within quotes denotes a string variable:

  `jsim:print_tree(user,"User")` → `jsim:print_tree(user,"Gianni")`

Some commands require the JANET Simulator application to be either running or stopped, and an in-depth description of each command along with its arguments and return values can be found in the *jsim* module ("apps/janet_simulator/src/jsim.erl").

The complete set of commands can also be printed directly in the *erl* shell via the `jsim:help()` command.

SIMULATION START AND STOP		
Command	**Description**	**Must run**
`jsim:run()`	Starts the JANET Simulator application	✗
`jsim:stop()`	Stops the JANET Simulator application	✓
`jsim:shutdown()`	Stops the JANET Simulator application and its ERTS	–

SIMULATION MONITORING

Print Database Tree		
Command	**Description**	**Must run**
`jsim:print_tree()`	Prints the database contents indented as a tree	–
`jsim:print_tree(user, "User")`	Prints the database contents of a specific user indented as a tree	–
`jsim:print_tree(loc,Loc_id)`	Prints the database contents of a specific location indented as a tree	–
`jsim:print_tree(sub,Subloc_id)`	Prints the database contents of a specific sublocation indented as a tree	–

Monitor Database Tree		
Command	**Description**	**Must run**
`jsim:monitor_tree()`	Prints the database contents indented as a tree every 10 seconds	–
`jsim:monitor_tree(PeriodSecs)`	Prints the database contents indented as a tree every *PeriodSecs* seconds	–
`jsim:demonitor_tree()`	Stops the periodic printing of the database contents indented as a tree	–

Print Running and Stopped Nodes		
Command	**Description**	**Must run**
`jsim:print_nodes()`	Prints a summary of running and stopped nodes	✔
`jsim:print_nodes(State)` `- State = running | stopped`	Prints a summary of running OR stopped nodes	✔

Print JANET Simulator Mnesia Tables		
Command	**Description**	**Must run**
`jsim:print_table()`	Prints the contents of all database tables	–
`jsim:print_table(SimTable)` `- SimTable = loc | sub | dev | sup | ctrmgr | devmgr`	Prints the contents of a specific database table	–

`jsim:print_record(SimTable,ID)`
`- SimTable = loc | sub | dev | sup | ctrmgr | devmgr`
`- ID = Loc_id | Sub_id | Dev_id`
	Prints a specific table record	–

Print JANET Controller Mnesia Tables		
Command	**Description**	**Must run**
`jsim:print_ctr_tree(Loc_id)`	Prints the contents of a controller's database indented as a tree	✔
`jsim:print_ctr_table(Loc_id)`	Prints all tables of a controller's database	✔
`jsim:print_ctr_table(Loc_id, CtrTable)` `- CtrTable = sub | dev`	Prints a specific table of a controller's database	✔

Remote Hosts Connectivity States		
Command	**Description**	**Must run**
`jsim:print_rem_hosts_states()`	Prints a summary of the connectivity states of the remote hosts used in the application	✔

JANET NODES START AND STOP

Per-Node Start and Stop		
Command	**Description**	**Must run**

`jsim:stop_node(NodeType,NodeID)`
`- NodeType = ctr | dev`
`- NodeID = Loc_id | Dev_id`
	Stops the controller or device node of the given NodeID	✔

`jsim:restart_node(NodeType,NodeID)`
`- NodeType = ctr | dev`
`- NodeID = Loc_id | Dev_id`
	Restarts the controller or device node of the given NodeID	✔

Per-Sublocation Start and Stop		
Command	**Description**	**Must run**
`jsim:stop_subloc(Sub_id)` `- Sub_id = {Loc_id,Subloc_id}`	Stops all device nodes in the given sublocation	✔
`jsim:restart_subloc(Sub_id)` `- Sub_id = {Loc_id,Subloc_id}`	Restarts all device nodes in the given sublocation	✔

Per-Location Start and Stop		
Command	**Description**	**Must run**
`jsim:stop_loc(Loc_id)`	Stops the controller and all device nodes in the given location	✔
`jsim:restart_loc(Loc_id)`	Restarts the controller and all device nodes in the given location	✔

All-Nodes Start and Stop		
Command	**Description**	**Must run**
`jsim:stop_all_nodes()`	Stops all controller and device nodes in the application	✔
`jsim:restart_all_nodes()`	Restarts all controller and device nodes in the application	✔

DATABASE MANIPULATION

--- WARNING: Using these commands WILL lead to inconsistencies with the remote database ---

Create		
Command	**Description**	**Must run**
`jsim:add_location(Loc_id, "Name",User,Port,"Hostname")`	Adds a new location, also starting its controller node if the application is running	–
`jsim:add_sublocation(Sub_id,"Name")` `- Sub_id = {Loc_id,Subloc_id}`	Adds a new sublocation in a location	–

`jsim:add_device(Dev_id, "Name",Sub_id,Type,"HostName")`
`- Sub_id = {Loc_id,Subloc_id}`
`- Type = fan | light | door | thermostat | conditioner`
	Adds a new device in a sublocation, also starting its node if the application is running	–

Update		
Command	**Description**	**Must run**
`jsim:dev_config_change(Dev_id,Config)` `- DevConfig = #devcfg record`	Changes a device's configuration	✔
`jsim:update_dev_subloc(Dev_id,Sub_id)` `- Sub_id = {Loc_id,Subloc_id}`	Changes a device's sublocation within its location	–
`jsim:update_loc_name(Loc_id,"Name")`	Updates a location's name	–
`jsim:update_subloc_name(Sub_id,"Name")` `- Sub_id = {Loc_id,Subloc_id}`	Updates a sublocation's name	–
`jsim:update_dev_name(Dev_id,"Name")`	Updates a device's name	–

Delete		
Command	**Description**	**Must run**
`jsim:delete_location(Loc_id)`	Deletes a location along with all its sublocations and devices	–
`jsim:delete_sublocation(Sub_id)` `- Sub_id = {Loc_id,Subloc_id}`	Deletes a sublocation, moving its devices to the default sublocation	–
`jsim:delete_device(Dev_id)`	Deletes a device	–

DATABASE BACKUP AND RESTORE

--- WARNING: Using these commands WILL lead to inconsistencies with the remote database ---

Backup		
Command	**Description**	**Must run**
`jsim:backup()`	Backs up the database contents to the "db/mnesia_backup.db" file	–
`jsim:backup("FileName")`	Backs up the database contents to "Filename" under the "db" directory	–

Restore		
Command	**Description**	**Must run**
`jsim:restore()`	Restores the database to the contents of the "db/mnesia_backup.db" file	✗
`jsim:restore("FileName")`	Restores the database to the contents of "FileName" under the "db" directory	✗

Clear		
Command	**Description**	**Must run**
`jsim:clear()`	Clears all database contents	✗