

Distributed Systems and Middleware Technologies  
JANET Home Simulator Installation Guide

Table of Contents

[0) Prerequisites 1](#_Toc85736846)

[1) Install Erlang/OTP 24.1 1](#_Toc85736847)

[2) Install Rebar3 2](#_Toc85736848)

[3) Configure the JANET Home Simulator 2](#_Toc85736849)

[4) Start the JANET Home Simulator 3](#_Toc85736850)

[5) Remote Nodes Hosts Configuration 4](#_Toc85736851)

JANET Home Simulator Installation Guide v1.0

1. Prerequisites
2. Linux operating system (*might* work on MacOs, not on Windows)
3. Availability of the *apt* package manager
4. Availability of *tar* or a similar archive manager
5. Install Erlang/OTP 24.1

Reference: [Building and Installing Erlang/OTP](https://erlang.org/doc/installation_guide/INSTALL.html)

Since the precompiled Erlang/OTP package in the *apt* repository is typically outdated, it must be compiled and installed from its source code.

1. Ensure that the minimal dependencies required to compile and build Erlang are installed

yourUser@yourHost~$ sudo apt-get install make libncurses5-dev gcc perl sed openssl flex

1. Download the Erlang/OTP 24.1 source code archive from [here](https://erlang.org/download/otp_src_24.1.tar.gz).

yourUser@yourHost~$ wget https://erlang.org/download/otp\_src\_24.1.tar.gz

1. Unpack the archive and move to its unpacked directory

yourUser@yourHost~$ tar -zxf otp\_src\_24.1.tar.gz  
yourUser@yourHost~$ cd otp\_src\_24.1

1. Set the $ERL\_TOP environment variable to the current directory

yourUser@yourHost~/otp\_src\_24.1$ export ERL\_TOP= `pwd` # NOTE: ` not ‘

1. Configure the Erlang installation

yourUser@yourHost~/otp\_src\_24.1$ ./configure

Note that the command will report that some optional libraries (such as *fop*, *xmlint*, *wxWidgets*, *jinterface*, *odbc*), but they are not required for running the JANET Simulator.

1. Build the Erlang binaries

yourUser@yourHost~/otp\_src\_24.1$ make -j NUM\_THREADS # -j NUM\_THREADS for parallel build

1. *(optional)* Validate the binaries by running a predefined test suite

yourUser@yourHost~/otp\_src\_24.1$ make release\_tests

yourUser@yourHost~/otp\_src\_24.1$ cd release/tests/test\_server

yourUser@yourHost~/otp\_src\_24.1/release/tests/test\_server$ $ERL\_TOP/bin/erl -s ts install -s ts smoke\_test batch -s init stop

Test results can be found in the “$ERL\_TOP/release/tests/test\_server/index.html” file (they should all be passed).

1. Install Erlang (might require root privileges depending on the default installation directory)

yourUser@yourHost~/otp\_src\_24.1$ [sudo] make install

1. Verify that Erlang was successfully installed and added to the $PATH via the *erl* command

yourUser@yourHost~$ erl

Erlang/OTP 24 [erts-12.1] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-threads:1] [jit]

Eshell V12.1 (abort with ^G)

1> \_

1. Install Rebar3

Reference: [Rebar3 – Getting Started](https://rebar3.readme.io/docs/getting-started)

Rebar3 is the de-facto standard project management tool for Erlang applications (akin to Maven for Java) which, by using a different project folder structure from the Erlang/OTP official one and by imposing a number of development constraints, often undocumented, represents a natural barrier for a more widespread adoption of the Erlang language.

1. Download the Rebar3 escript from [here](https://s3.amazonaws.com/rebar3/rebar3).

yourUser@yourHost~$ wget https://s3.amazonaws.com/rebar3/rebar3

1. Install Rebar3

yourUser@yourHost~$ ./rebar3 local install

===> Extracting rebar3 libs to ~/.cache/rebar3/lib...

===> Writing rebar3 run script ~/.cache/rebar3/bin/rebar3...

===> Add to $PATH for use: export PATH=$PATH:~/.cache/rebar3/bin

1. Permanently add the Rebar3 local installation folder (“~/.cache/rebar3/bin”) to the $PATH

yourUser@yourHost~$ vi .bashrc  
…

export PATH=$HOME/.cache/rebar3/bin:…:$PATH

1. Verify that Rebar3 has been correctly installed and added to the $PATH

yourUser@yourHost~$ rebar3

Rebar3 is a tool for working with Erlang projects.

Set the environment variable DEBUG=1 for detailed output.

Usage: rebar3 [-h] [-v] [<task>]

…

1. Configure the JANET Home Simulator

The JANET Home Simulator can be configured by editing the “JANET Simulator Public Configuration Parameters” in the “config/sys.config” file, which are summarized here:

|  |  |  |  |
| --- | --- | --- | --- |
| Configuration Parameter | Description | Allowed Values | Note |
| sim\_rest\_port | The OS port to be used by the JANET Simulator REST server | integer() > 0 | - |
| remote\_rest\_server\_addr | The address of the remote server accepting REST requests from JANET controller nodes | list() / string() | If deployed on the same host, the machine name (e.g. “yourHost”) must be used over “localhost” |
| remote\_rest\_server\_port | The port of the remote server accepting REST requests from the JANET Controller nodes | integer() > 0 | - |
| nodes\_hosts | The list of hostnames JANET nodes can be deployed in | [list() / string()] | If the JANET Simulator host is to be included, use the full machine name (e.g. “yourHost” over “localhost”) |

1. Start the JANET Home Simulator

The JANET Simulator Erlang Run-Time System (ERTS) can be started by executing the “start” bash script located in the “janet\_home\_simulator” folder, which will automatically fetch all application dependencies, compile them, and start the virtual machine and a set of application components (not including the JANET Simulator application itself).

yourUser@yourHost~/janet\_home\_simulator$ ./start

===> Verifying dependencies...

===> Analyzing applications...

===> Compiling ranch

===> Compiling cowlib

===> Compiling jsone

===> Compiling gun

===> Compiling cowboy

===> Analyzing applications...

===> Compiling janet\_simulator

===> Compiling janet\_controller

===> Compiling janet\_device

Erlang/OTP 24 [erts-12.1] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-threads:30] [jit]

Eshell V12.1 (abort with ^G)

(janet-simulator@yourHost)1> ===> Booted mnesia

\_

The JANET Simulator application itself can be started with the “jsim:run()” command, which in its first execution will install the Mnesia database used by the application (which will initially be empty).

Erlang/OTP 24 [erts-12.1] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-threads:30] [jit]

Eshell V12.1 (abort with ^G)

(janet-simulator@yourHost)1> ===> Booted mnesia

(janet-simulator@yourHost)1> jsim:run().

The Mnesia database is not installed, installing it now

JANET Simulator Mnesia database successfully installed

[locs\_init]: <WARNING> No location is present in the database, no location tree will be started

ok

\_

The JANET Simulator application and ERTS can be stopped respectively via the “jsim:stop()” and “jsim:shutdown()” commands, with the complete set of supported commands being outlined in the JANET Home Simulator User Guide.

(janet-simulator@yourHost)2> jsim:stop().

JANET Simulator stopped

ok

(janet-simulator@yourHost)3> jsim:shutdown().

ok

(janet-simulator@yourHost)4> yourUser@yourHost~/janet\_home\_simulator$

1. Remote Nodes Hosts Configuration

The remote hosts JANET nodes can be deployed in (defined in the “nodes\_hosts” configuration parameter) must have Erlang/OTP 24.1 installed (step 1) and require the following configuration steps:

1. Ensure the DNS reachability between the JANET Simulator and all remote hosts nodes.

This can be obtained through the use of a DNS server or by manually mapping each host to its IP in the “/etc/hosts” files

yourUser@yourHost1~$ sudo vi /etc/hosts

192.168.0.1 yourHost1 # localhost

192.168.0.2 yourHost2

yourUser@yourHost2~$ sudo vi /etc/hosts

192.168.0.1 yourHost1

192.168.0.2 yourHost2 # localhost

1. Enable the JANET Simulator host to establish an *ssh* connection to each nodes host without being prompted for a password.

This can be obtained via the following commands, with a more comprehensive guide being available [here](https://www.thegeekstuff.com/2008/11/3-steps-to-perform-ssh-login-without-password-using-ssh-keygen-ssh-copy-id/).

yourUser@JANETSimulatorHost~$ cd .ssh

yourUser@JANETSimulatorHost~/.ssh$ ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/home/yourUser/.ssh/id\_rsa): # Press Enter

Enter passphrase (empty for no passphrase): # Press Enter

Enter same passphrase again: [Pess enter key] # Press Enter

Your identification has been saved in /home/yourUser/.ssh/id\_rsa.

Your public key has been saved in /home/yourUser/.ssh/id\_rsa.pub.

The key fingerprint is:

33:b3:fe:af:95:95:18:11:31:d5:de:96:2f:f2:35:f9

yourUser@JANETSimulatorHost~/.ssh$ ssh-copy-id nodeHost1

yourUser@nodeHost1’s password: •••••• # Enter the “yourUser” password on “nodeHost1”

Number of key(s) added: 1

Now try logging into the machine, with : “ssh ‘nodeHost1’”

And check to make sure that only the key(s) you wanted were added.

yourUser@JANETSimulatorHost~/.ssh$ ssh nodeHost1

yourUser@nodeHost1~$ \_

1. Copy the folders containing the resource file and compiled bytecode of all the JANET Home Simulator application dependencies (“\_build/default/lib”) into the “lib” directory in the Erlang installation folder (default: “/usr/lib/erlang/lib”) (might require root privileges).

yourUser@yourHost~/janet\_home\_simulator$ [sudo] cp -r \_build/default/lib/cowlib \_build/default/lib/cowboy \_build/default/lib/gun \_build/default/lib/ranch \_build/default/lib/jsone \_build/default/lib/janet\_simulator \_build/default/lib/janet\_controller \_build/default/lib/janet\_device /usr/lib/erlang/lib

yourUser@yourHost~/janet\_home\_simulator$