**Version 0.5.0**

# BUILD INSTRUCTIONS FOR RVI

This document describes the build process for the RVI project on an Ubuntu 14.04 Linux machine.

Please see `README.md` for a general description of the project and its structure.

Please see `CONFIGURE.md` for details on configuring and launching the system once it has been built.

The first milestone of the RVI project is the HVAC demo. Please see `hvac_demo/README.md` for details on how to setup, launch and drive the demo.

# READER ASSUMPTIONS

In order to build the system, the reader is assumed to be able to:

1. Have a basic understanding of Linux system operations.
2. Install packages on the system.

Please note that the configuraiton process, described in `CONFIGURE.md` may have additional skill requirements.

# PREREQUISITES

1. The Ubuntu 14.04 system have the latest updates installed.
2. The user can gain root access to install packages.
3. There is at least 5GB of space availabled for packages and code.

---

# INSTALLATION PROCESS

## INSTALL GIT

Use `apt-get` to install git, which is used to access the Automotive Grade Linux repositories where the code resides:

```
sudo apt-get git
```

## INSTALL ERLANG

Install Erlang 18.2, or a later version 18 release:

Tested packages of the latest versions of Erlang can be downloaded from [packages.erlang-solutions.com](packages.erlang-solutions.com)

Add the following line to your /etc/apt/sources.list

```
deb http://packages.erlang-solutions.com/ubuntu trusty contrib
```

Update and install erlang

```
sudo apt-get update
sudo apt-get install erlang
```

## CLONE THE RVI REPOSITORY

Use the newly installed `git` tool to clone (copy) the RVI repository to the build system.

```
git clone https://github.com/PDXostc/rvi_core.git
```

The clone will be downloaded into a newly created `rvi_core` subdirectory.

---

## BUILD THE RVI SYSTEM

Run `make` to build the dependency code in `deps` and the top level project in the `rvi` directory.

```
make compile
```

The local `rebar` command is used to retrieve the dependencies. See `rebar.config` and `deps/*/rebar.config` for a list of dependencies.

See the rebar project for a detailed description of the rebar Erlang build tool.

```
Expected output:
$ make
./rebar get-deps
==> goldrush (get-deps)
==> lager (get-deps)
==> src (get-deps)
==> ale (get-deps)
==> src (get-deps)
...
./rebar  compile
==> goldrush (compile)
Compiled src/glc.erl
Compiled src/glc_lib.erl
Compiled src/glc_code.erl
...
/.../rvi_core/deps/exo/src/exo_ssh.erl:18: Warning: undefined callback function
code_change/3 (behaviour 'ssh_channel')
/.../rvi_core/deps/exo/src/exo_ssh.erl:18: Warning: undefined callback function
handle_call/3 (behaviour 'ssh_channel')
...
cp deps/setup/setup_gen scripts/
(cd components/authorize && make escript)
ERL_LIBS=/.../rvi_core/components/authorize/..:/..../jlr/rvi_core/components/authorize/../.
./deps ./rebar escriptize
==> authorize (escriptize)
cp components/authorize/author scripts/
$
```

Some warnings are expected, and are usually not a failure indication:

The compiled code is available under `ebin/`, `components/*/ebin` and `deps/*/ebin`.

# CREATE A RELEASE

See `CONFIGURE.md` for details on configuring and creating a developer and production release that can be launched.