

TuCSon "how-to"

In this brief "how-to", you will learn how to get TuCSon, build it and run examples showcasing its features.

Assumptions are you are familiar with Java (compilation), Git (cloning repositories) and, optionally, ANT (buildfiles).

1. [Getting Started with TuCSon](#)
 - 1.1 [Downloading](#)
 - 1.2 [Compiling](#)
 - 1.3 [Deploying](#)
 - 1.4 [Running](#)
 2. [Contact Information](#)
-

1. Getting Started with TuCSon

1.1 Downloading

If you want the *ready-to-use* distribution of TuCSon, download **tucson.jar** archive from the "Downloads" section, here > <http://bitbucket.org/smariani/tucson/downloads>. **TuCSon_boot.sh** bash script is also available for quick boot of TuSCoN components and examples.

If you want the *source code* of TuCSon, clone the TuCSon **Git repository** hosted here on Bitbucket, at > <http://smariani@bitbucket.org/smariani/tucson.git> (e.g., from a command prompt type `$> git clone https://smariani@bitbucket.org/smariani/tucson.git`) [1].

In the former case, skip to the "[Running](#)" section below. In the latter case, keep reading.

1.2 Compiling

By cloning TuCSon you have downloaded a folder named **tucson/**, with the following *directory structure*:

```
tucson/  
|__...  
|__TuCSon/  
|   |__...  
|   |__ant-scripts/  
|       |__build.xml  
|       |__environment.properties  
|   |__eclipse-config/  
|   |__how-to/  
|   |__license-info/  
|   |__trunk/  
|       |__src/
```

TuCSoN depends on 1 other Java library to function properly [2]:

- **tuProlog**, downloadable from tuProlog "Download" page, here > <http://apice.unibo.it/xwiki/bin/view/Tuprolog/Download> (2p.jar)

Once you got the above libraries, you are ready to compile TuCSoN source code.

The easiest way to do so is by exploiting the **ANT** script named `build.xml` within folder `ant-scripts/`, which takes care of the whole building process for you, from compilation to deployment (covered in next section). To do so, you need to have ANT installed on your machine [3]. If you don't want to use ANT, build TuCSoN jar archive using the tools you prefer, then skip to the "Running" section below.

To compile TuCSoN using ANT:

1. Edit the `environment.properties` file according to your system configuration:
 - 1.1 Tell ANT where your JDK and your `java` tool are
 - 1.2 Tell ANT which libraries are needed to compile TuCSoN (the one you just downloaded, that is tuProlog)
 - 1.3 Tell ANT where you put such libraries (e.g. if you put them into `TuCSoN/libs/` you are already set)
 - [1.4 Tell ANT your Bitbucket username (for automatic syncing with TuCSoN repository, **not supported at the moment**)]*
2. Launch the ANT script using target `compile` (e.g., from a command prompt position yourself into the `ant-scripts/` folder then type `$> ant compile`) [4]. This will create folder `classes/` within folder `TuCSoN/` and therein store Java `.class` files.

Other ANT targets are available through the `build.xml` file: to learn which, launch the ANT script using target `help`.

1.3 Deploying

Deploying TuCSoN is as simple as giving a different build target to the ANT script `build.xml`:

- if you only want the **TuCSoN jar** archive, ready to be included in your Java project, launch the script using target `lib`. This will compile TuCSoN source code into binaries (put into `TuCSoN/classes/` folder) then package them to **tucson.jar** into `TuCSoN/lib/` folder [5].
- if you want a **ready-to-release distribution** of TuCSoN, including also documentation and support libraries, launch the script using target `dist`. This will:
 - compile TuCSoN source code into binaries, put into `TuCSoN/classes/` folder
 - package them to `TuCSoN-1.11.0.0209.jar`, put into `TuCSoN/lib/` folder
 - generate Javadoc information, put into `TuCSoN/doc/` folder
 - create folder `rel/TuCSoN-${version}` including:
 - folder `docs/` including the generated Javadoc information as well as this "how-to"
 - folder `libs/` including tuProlog library used to build TuCSoN
 - folder `rel/` including TuCSoN jar archives

The complete directory structure obtained by launching `ant dist` build process should look like the following (assuming you put tuProlog library in folder `TuCSon/libs/`):

```
tucson/
|__...
|__TuCSon/
|   |__...
|   |__ant-scripts/
|   |   |__build.xml
|   |   |__environment.properties
|   |__classes/
|   |__doc/
|   |__eclipse-config/
|   |__how-to/
|   |__rel/
|   |   |__TuCSon-${version}/
|   |   |   |__docs/
|   |   |   |   |__how-to/
|   |   |   |   |__javadoc/
|   |   |   |__libs/
|   |   |   |__rel/
|   |__...
|__lib/
|__libs/
|__license-info/
|__trunk/
|__src/
```

Other ANT targets are available through the `build.xml` file: to learn which, launch the ANT script using target `help`.

1.4 Running

To run TuCSon, you need:

- TuCSon jar, **tucson.jar**
- tuProlog jar, **2p.jar**

Supposing you built TuCSon using the provided ANT script [\[6\]](#) and that you are comfortable with using a command prompt to launch Java applications [\[7\]](#):

1. open a command prompt and position yourself into either `TuCSon/lib/` or `TuCSon/rel/TuCSon-${version}/rel/` folder
2. launch the TuCSon Node service, e.g. as follows [\[8\]](#):

```
java -cp tucson.jar:../libs/2p.jar alice.tucson.service.TucsonNodeService
```

The TuCSon ASCII logo on the command prompt, as depicted below.

[4] If you are using [Eclipse IDE](#) for developing in Java, ANT is included: click "Window > Show View > Ant" then click "Add buildfiles" from the ANT view and select file `build.xml` within `ant-scripts/` folder. Now expand the "TuCSon build file" from the ANT view and finally double click on target `compile` to start the build process.

[5] Actually, also a **tucson-noexamples.jar** is built. It is the same as **tucson.jar** except for the explanatory examples in package `alice.tucson.examples.*`, which are excluded.

[6] If you directly downloaded TuCSon jar or if you built it from sources without using the provided ANT script, simply adjust the given command to suit your configuration.

[7] If you do not want to use the command prompt to launch Java applications, adjust the given command to suit your configuration, e.g., if you are using [Eclipse IDE](#): right-click on "tucson.jar > Run As > Run Configurations..." then double-click on "Java Application", finally select "TucsonNodeService - alice.tucson.service" as the main class (`-cp tucson.jar:../libs/2p.jar` is automatically added by Eclipse according to project's build path settings).

[8] Separator `:` works on Mac & Linux only, use `;` on Windows.

[9] Actually, a TuCSon agent identifier can be any valid tuProlog *ground term*. See tuProlog documentation, [here](#).
