Stood

Installation Manual

Revision G

Ellidiss Technologies
http://www.ellidiss.com

1 Install the software

This section explains how to install **Stood 5.5** on your system. This version of the software is a 32 bits application that may run on a **Windows** or **Linux** platform, but the software must be installed on a local or a remote file system first. Following configurations are supported:

- local installation on a standalone **Windows PC**.
- network installation on an **Windows** server, for **Windows** only client platforms.
- local installation on a standalone **Linux** platform.
- network installation on an **Linux** server for **Windows** and **Linux** client platforms.

There are no strong constraints as regards to the location of the software on a **Linux** file system. The installation procedure does not require privileged rights unless you are accessing protected directories. Here are a few recommendations:

- `bin.xxx`, `sbprolog` and examples directories should be stored in an area shared by all the users of **Stood**.
- `config` directory should be stored in an area shared by all the members of a same **Project**.
- `stood.ini` or `.stoodrc` files may be customized by each user and located in their own working directories.

The intallation program for **Windows** uses the default following locations:

- `Program Files (x86)` for the binaries, configuration and documentation files. Access to this directory requires administration rights.
- `User Documents` directory for the examples, so that they can be edited without administration rights.

2 Distribution media

The software may be delivered in several ways:

- download from the Internet
- on a **CD-Rom** or a **USB** stick

If **Internet** access is available on your system, you can download **Stood** from the main **Ellidiss** web site:

```
http://www.ellidiss.com
```

Additional information about Stood is available on the technical support web site:

```
http://www.ellidiss.fr
```

Otherwise, if you have received a **CD-Rom** or a **USB** stick, the required installation files should be directly copied from this disk, after it has been properly mounted on your system.

For any other questions or information, please contact the **Stood** technical support using one of the following email addresses:

info@ellidiss.com
stood@ellidiss.fr

2.1 Distribution contents

The archive files that come with the **Stood** distribution are named as follow:

- `Stood550Windows.exe`: auto extractible setup file to install **Stood** locally on a **PC** running **Windows XP**, **Vista**, **Windows 7**, **Windows 8** or **Windows 10**.
- `Stood551Linux.tar.gz`: compressed archive containing the distribution files for **PC Linux** platforms.

The following chapters describe the installation procedure to be followed after the appropriate components have been copied to your system. Once installed (i.e. having executed the installation program under **Windows** or uncompressed the installation archive under **Linux**), a typical installed distribution contains the following directories:

- `bin.w32`: binary files for **PC Windows**
- `bin.l32`: binary files for **PC Linux**
- `config` : **HOOD** configuration files for all platforms
- `config_AADL` : **AADL** configuration files for all platforms
- `sbprolog` : prolog engine and libraries used by **LMP** tools
- `bash` : a **Unix** shell subset, required for **Windows** installation only
- `examples` : a few **HOOD** design examples for **Ada**, **C** and **C++**
- `examples_AADL` : a few examples of **AADL** projects
- `examples_libs` : a few libraries to be used with the **HOOD** examples
- `documentation` : manuals

Note that the companion product **AADL Inspector** is no more included into the **Stood** distribution package. It must be installed separately, but can still be launched automatically by the **AADL** code generator.

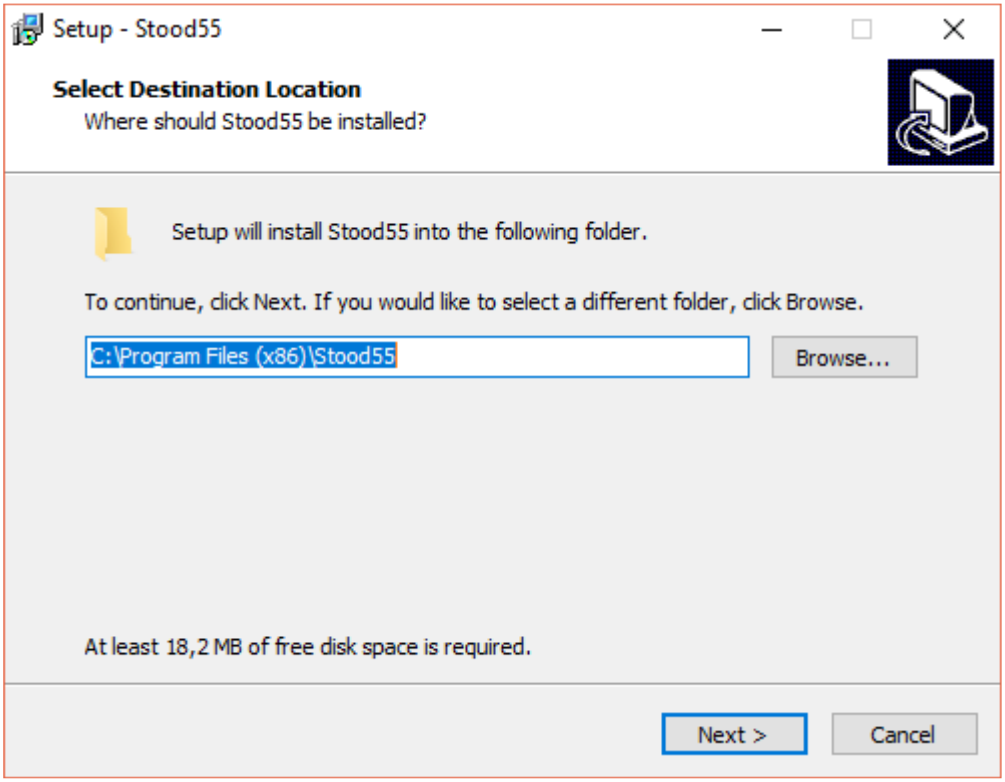
3 Installation on a Windows PC

WARNING: Before installing **Stood** on **Windows**, please take care to uninstall any previously installed version of the product on the **PC**.

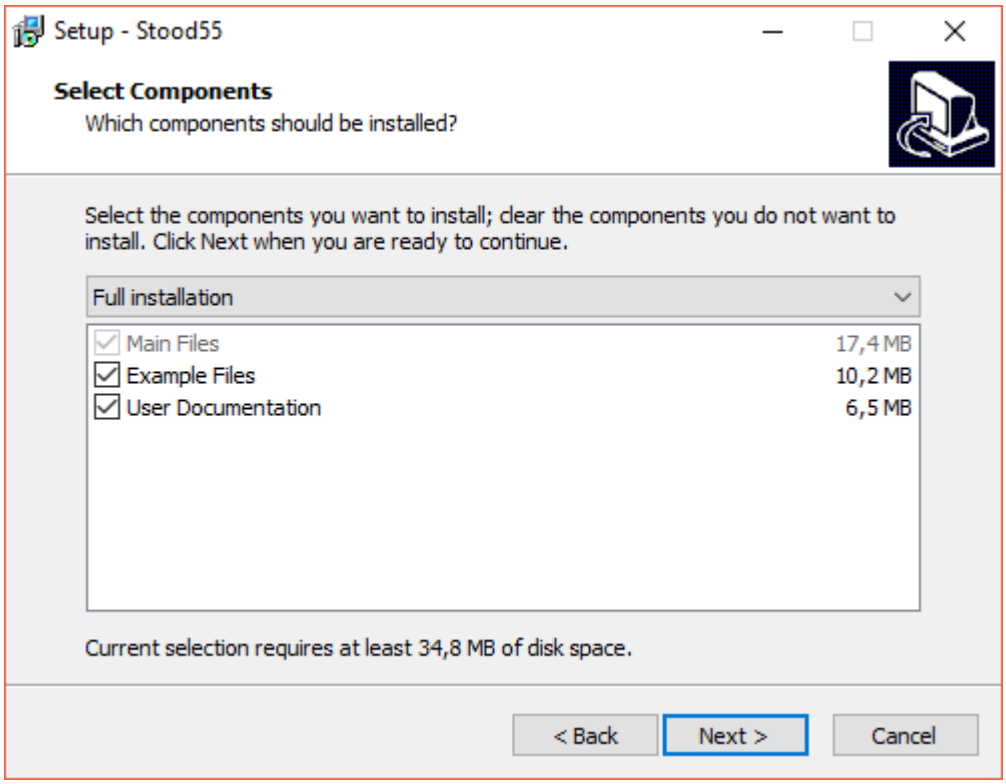
To install **Stood** on a **Windows** network, or a standalone **Windows PC**, execute the following auto extractible archive file:

```
Stood550Windows.exe
```

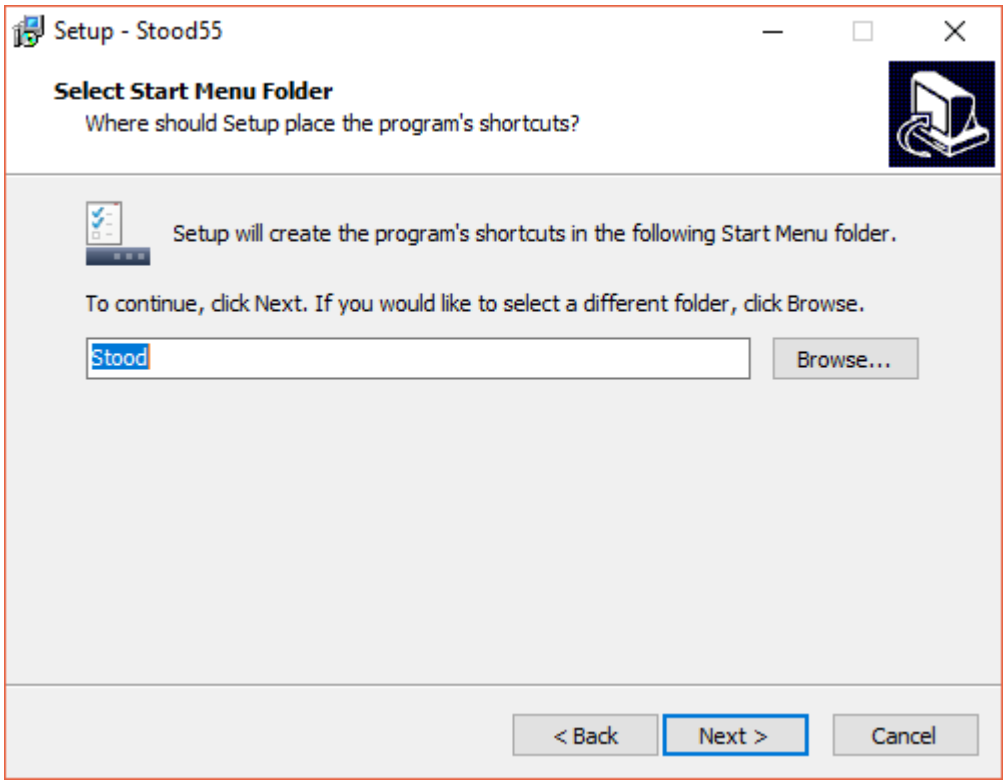
Note that you may need administration rights and be asked for confirmation to enable this program to make changes to your system. A setup dialog window should appear. Please follow the instructions to go forwards with the installation procedure.



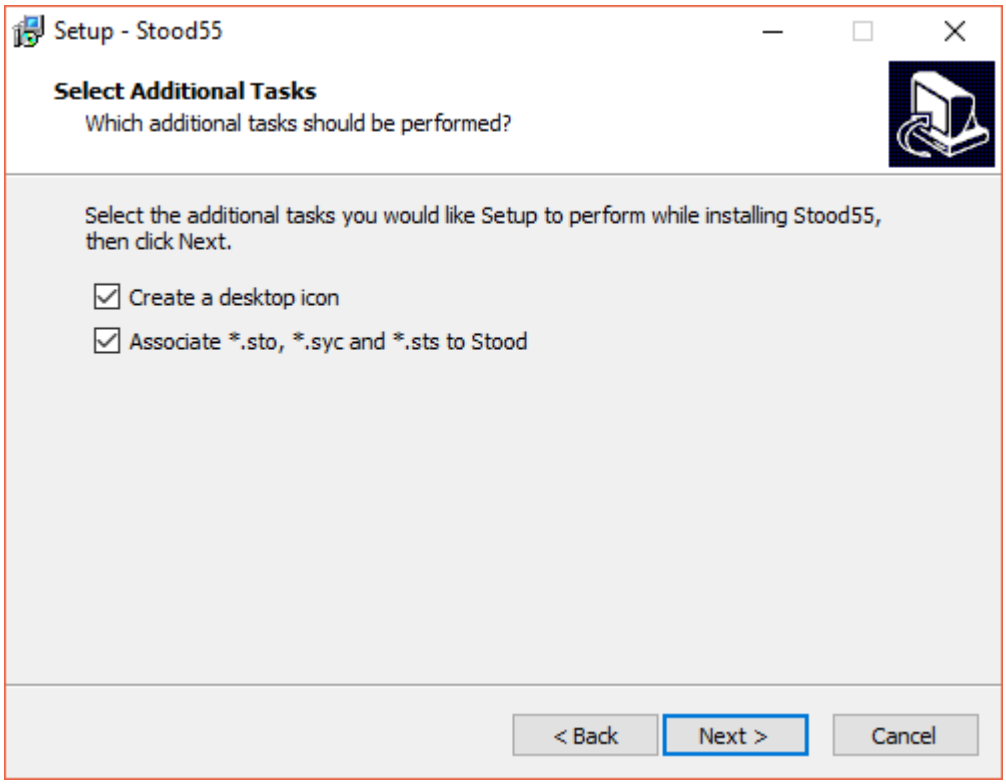
The distribution includes optional packages containing a few examples and user documentation. These examples are not mandatory and do not affect the use of **Stood**. They can be removed from the list to save disk space.

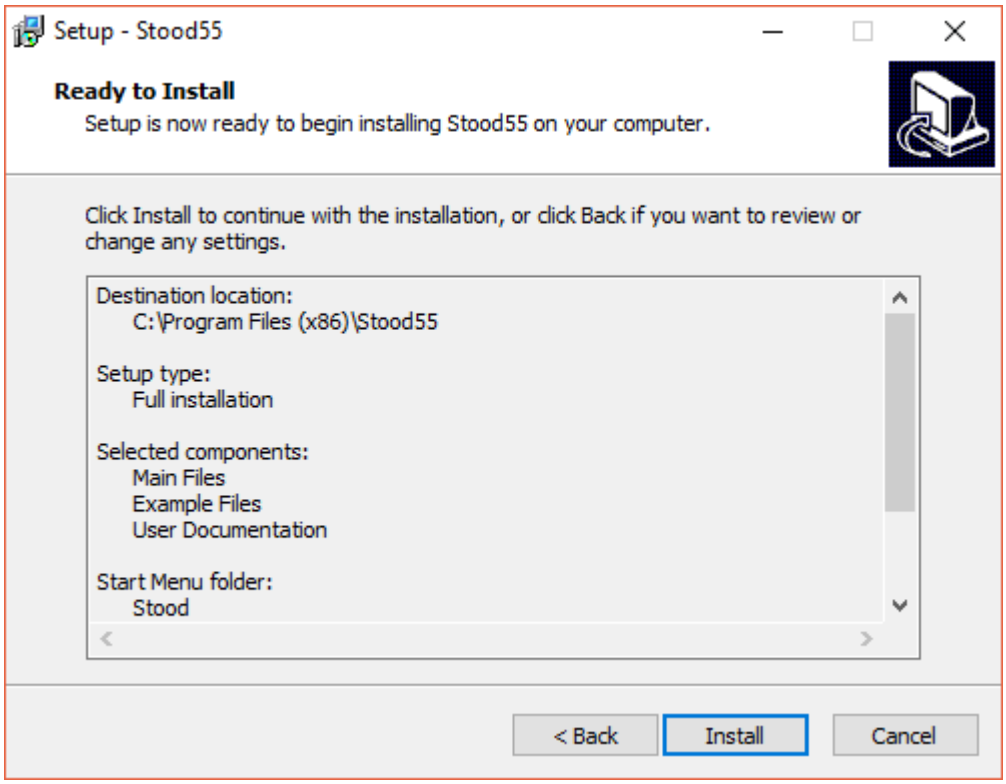


You can select the start menu folder into which the new program and documentation shortcuts will be added.

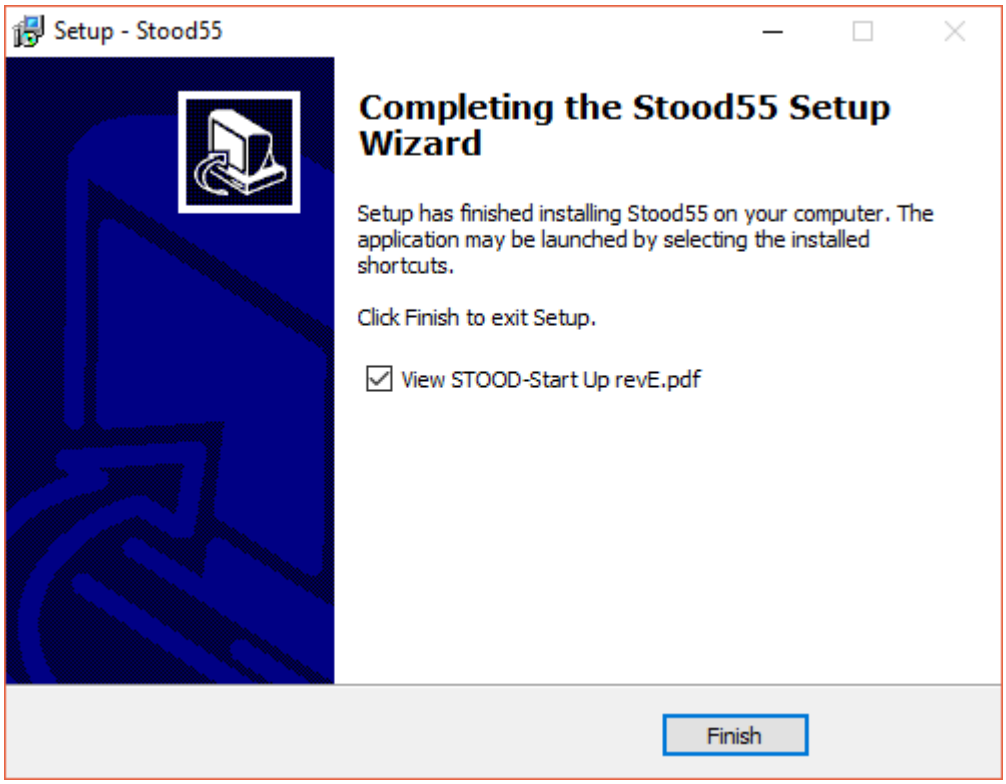


A few **Windows** file types may be associated to **Stood** during the installation process. When double-clicking on them, the following action will occur: `*.syc` opens **Stood** and loads the corresponding **Project**. `*.sto` opens **Stood** and loads the **Design** of selected `Stood.sto`. `*.sts` opens **Stood** and executes the corresponding **STShell** script.





Clicking on the install button will complete the installation procedure.



Finally, the quick start-up note can be opened after the product is properly installed on your system.

3.1 Installation on a Linux computer

To install **Stood** on a **Linux** computer, you need to get a copy of the installation archive file:

```
Stood551Linux.tar.gz
```

Change directory so that you are where **Stood** has to be installed. Super user rights are not required, except if the software has to be installed in a protected directory:

```
$ cd installation_directory
```

Then, copy, uncompress and expand the archive file as follow:

```
$ gunzip Stood551Linux.tar.gz
$ tar xvf Stood551Linux.tar
```

Around 35 Megabytes of free disk space is required on the server.

Depending on the actual configuration of the **Linux** distribution, the following additional actions may be required:

- Installing the 32 bits compatibility package (`ia32-libs`)
- Installing the **OpenMotif** package (`libXm.so`)
- Installing `csh`
- Installing **X11** fonts (75 dpi and 100 dpi)

As soon as all the transferred files are expanded, the installation is completed. **Stood** may be launched in demonstration mode without any further action, by running:

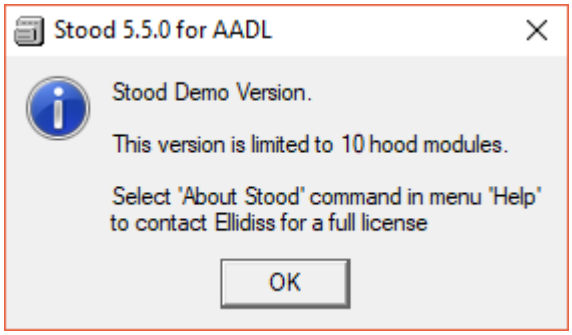
```
$ bin.l32/stood
```

4 Install the license key

Except for the demonstration mode, **Stood** requires a proper user license and its associated software key to operate properly. Several license management systems are available.

4.1 Demonstration mode

If no license key has been installed (default configuration), or if the number of used sharable tokens is exceeded, **Stood** runs in demonstration mode. In this case, you are limited to 10 **Components**.



Please note that the demonstration mode should be restricted to discovery and educational purposes, else it denotes the lack of a proper user license for the software, and no guarantee is given by **Ellidiss** while using **Stood** in demonstration mode.

4.2 Time-limited licenses

A time-limited software license key may be installed in order to enable a fully featured use of **Stood** until a given date. This software license key must be inserted inside the `stood.ini` (for **Windows** platforms) or `.stoodrc` (for **Unix** platforms) initialization file.

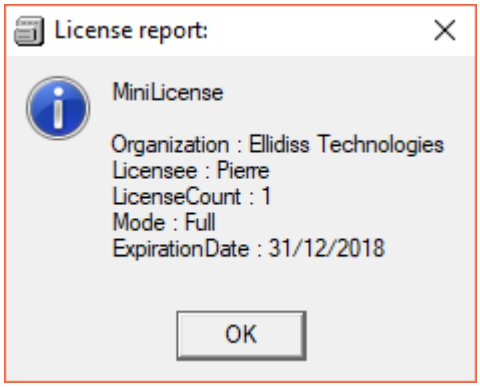
On a **Windows PC**, the time-limited license key information looks like the following:

```
[Licensing]
Organization=Company_Name
Licensee=User_Name
LicenseCount=1
Mode=Full
ExpirationDate=31/12/2018
Password=1234567
```

On a **Linux PC**, the same time-limited license key information looks like the following:

```
Licensing.Organization:Company_Name
Licensing.Licensee:User_Name
Licensing.LicenseCount:1
Licensing.Mode:Full
Licensing.ExpirationDate:31/12/2018
Licensing.Password:1234567
```

When **Stood** is running, information about actually used license tokens may be obtained with *Help/About license...* menu of the main window:



4.3 Network shared license file (NFL)

A license file may be installed on a **Windows** or **Linux** network server, in order to make floating license tokens available to any connected client. For this system to work properly, the remote server file must be visible from each client platform. The tool administrator must contact **Ellidiss** technical support to install a floating license tokens file on a network server, and users must add the following information inside the `stood.ini` (for **Windows** plartforms) or `.stoodrc` (for **Linux** platforms) initialization file on each client platform.

On a **Windows PC**, the network floating license key information looks as follows:

```
[NFL]
File=\\server\license\stood.nfl
ReleaseDelay=1440
```

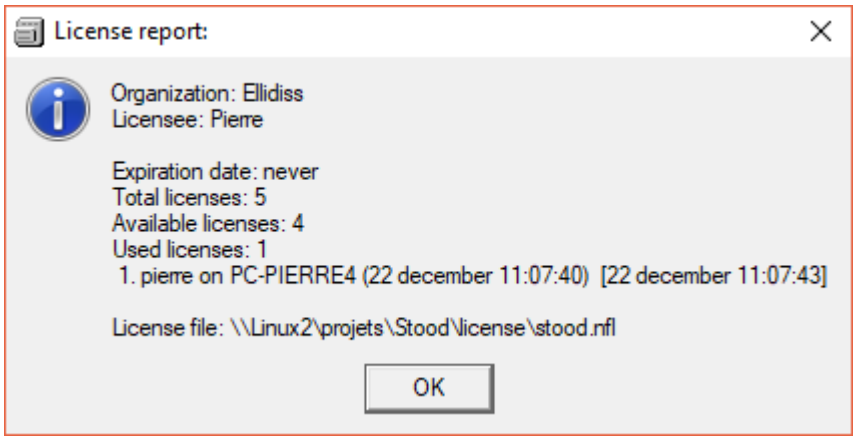
Note that the specified directory must be actually shared on the network. For instance, check that the two options "*Share this folder on the network*" and "*Allow network users to change my files*" are set in the properties of the directory. The way to configure these options may change according to the version of **Windows** you are using.

On a **Linux** workstation, the same network floating license key information looks as follows:

```
NFL.File://server/license/stood.nfl
NFL.ReleaseDelay:15
```

Note that the specified directory must be properly mounted on the **NFS** server in order to be visible across the network. In addition, please check that both the directory and the `.nflfile` have read and write access for any **Stood** user.

When **Stood** is running, information about actually used license tokens may be obtained with *Help/About license...* menu of the main window:



4.4 Network license server (ETFL)

Since version 5.5 of **Stood**, it is also possible to get a license token across the network if the ET Floating License (**ETFL**) server program has been installed on one of the connected computers. If this is the case, you simply need to know the following information and update the stood initialization file accordingly:

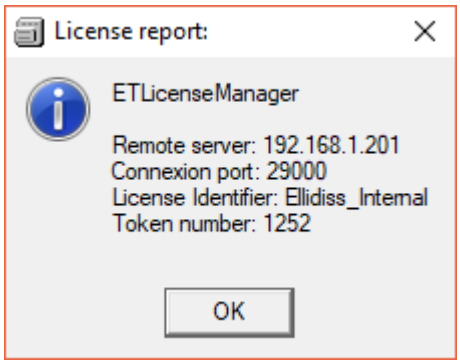
On **Windows** (`stood.ini`):

```
[ETLicense]
ServerName=192.168.1.201
ServerPort=29000
Id=Ellidiss_Internal
```

On **Linux** (`.stoodrc`):

```
ETLicense.ServerName:192.168.1.201
ETLicense.ServerPort:29000
ETLicense.Id:Ellidiss_Internal
```

When **Stood** is running, information about actually used license tokens may be obtained with *Help/About license...* menu of the main window:



Note that the installation procedure of the **ETFL** server program is provided in a separate manual.

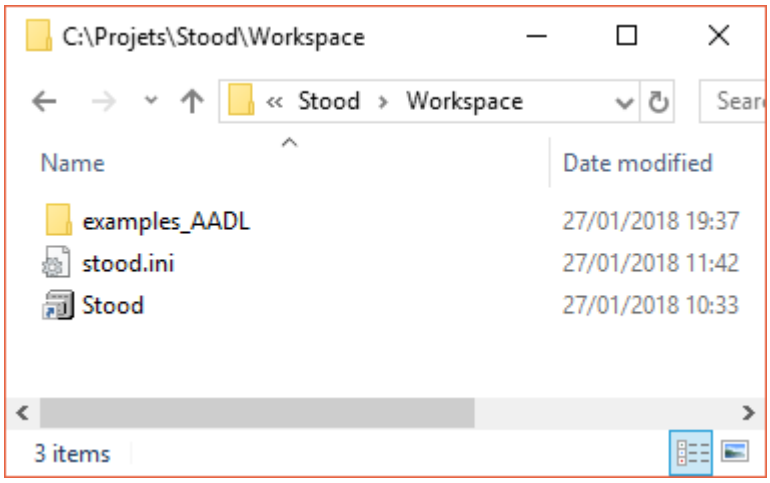
5 Quick start guide

After installing the software and its license, **Stood** is ready to use, and all its features for **AADL** or **HOOD** architectural and detailed design activities are enabled. This document provides some basic information on using the tool so that you can get started.

This quick start guide assumes that you are using a standard standalone installation on a **Windows PC**. Please note that there may be a few differences if you are using a **Unix** version, or a network installation.

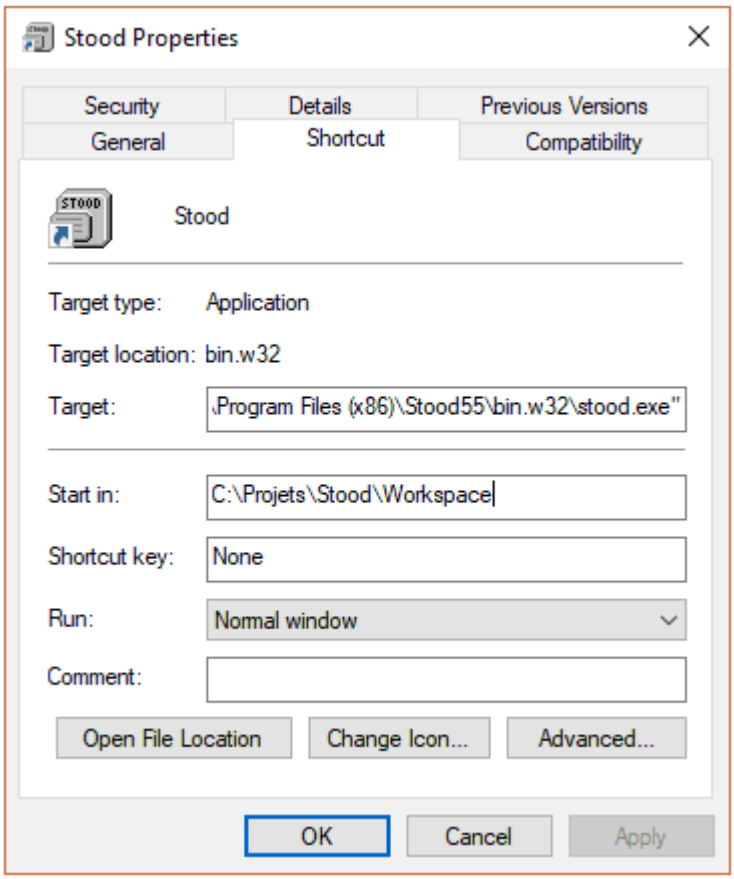
5.1 Launch Stood

The standard installation creates two shortcuts for **Stood** in the **Windows** startup menu (*Stood for HOOD* and *Stood for AADL*), and one on the desktop (*Stood for AADL*). The default working directory (workspace) that is defined by the installation program is `{Current_User_Directory}\Documents\Stood`. This default workspace contains a set of predefined examples that are described at chapter 2.



You can create other working directories anywhere else on your system. To do so, create a new directory, ensure that you have all the necessary access rights within this directory, and add the following items into it:

- A shortcut of the **Stood** executable (*Stood for HOOD* or *Stood for AADL*).
- A local initialization file (*stood.ini*).
- Optionally, a copy of the predefined examples.



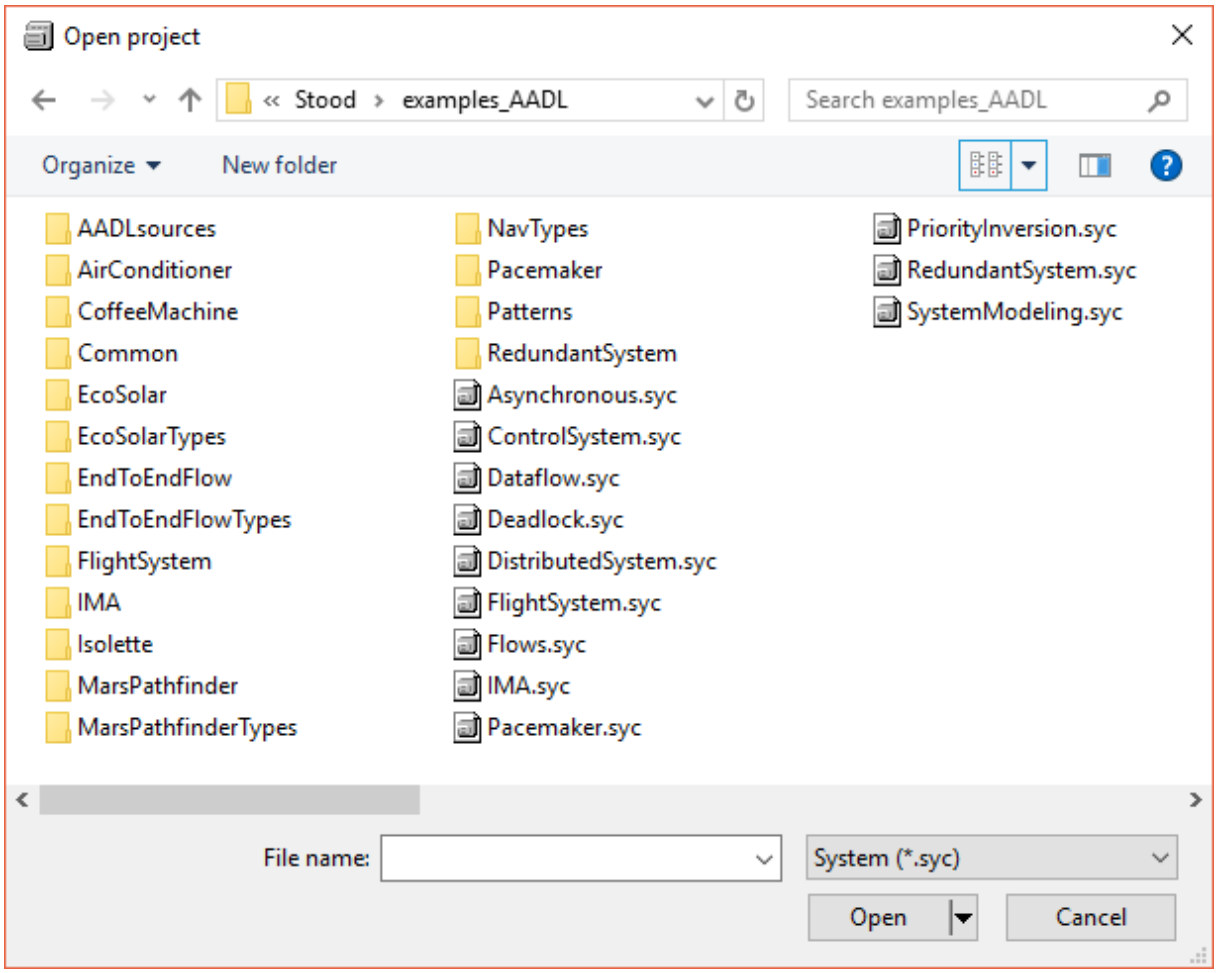
The local initialization file (`stood.ini`) may be created and edited with any text editor. This file should only contain the properties that differ from the standard configuration. These properties will overload the default ones if **Stood** is executed from this working directory. For instance, this local `.ini` file may typically contain the following customized sections:

```
[General]
Welcome=My Stood workspace

[Environment]
PROJECT=My_Project
COMPANY=My_Company
OSATE_PATH=/cygdrive/C/Osate/osate.exe
AI_PATH=/cygdrive/C/AI-1.6/bin/AADLInspector.exe

[ConfigurationManagement]
Versions=Draft,SRR,PDR,CDR,AR

[ETLicense]
ServerName=192.168.1.201
ServerPort=29000
Id=Ellidiss
```

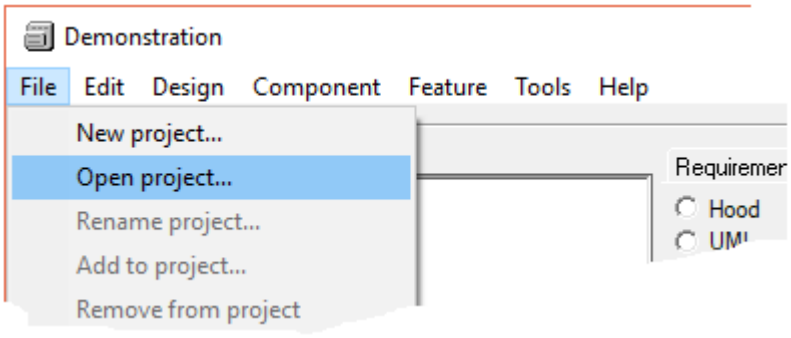
5.2 View the examples

To view the predefined examples that are stored in the default workspace, launch **Stood** from the Start Up menu or the shortcut that was created on the Desktop during installation. If you want to view the predefined examples from within another workspace, please copy the example directories into this location first and launch **Stood** with the shortcut you created in this workspace.

Each example is identified by a `.syc` file that defines the list of **Designs** that collaborate to a given **Project**. A same **Design** may collaborate to several **Projects**.



To load one of the examples, use the *Open project*... option in the *File* menu:



This opens a standard file selector. Navigate to the `examples` directory and select a `.syc` file. This will load the corresponding example. The directories `examples` and `examples_libs` contain **HOOD-Ada**, **HOOD-C/C++** designs and libraries, whereas the directory `examples_AADL` contains ten illustrative examples of graphical **AADL** models. For each of them, the complete set of textual **AADL** code can be generated and analysed with the **AADL Inspector** companion tool.

Example 1. "Dataflow.syc": dataflow inter-threads communication
in AADL Inspector, check the effect of sampled, immediate or delayed connections

Example 2. "Deadlock.syc": possible deadlock situation
in AADL Inspector, check the effect of using Priority Ceiling Protocol or not

Example 3. "PriorityInversion.syc": possible priority inversion situation
in AADL Inspector, check the effect of using Priority Ceiling Protocol or not

Example 4. "Asynchronous.syc": user interaction with a simulated coffee machine
in AADL Inspector, use the process I/O control panel

Example 5. "DistributedSystem.syc": CAN bus based distributed architecture of a solar vehicule
in AADL Inspector, observe the impact of the bus communication on threads scheduling

Example 6. "ControlSystem.syc": an automated heat regulator
in AADL Inspector, specify a desired temperature and observe the behavior of the

regulator

Example 7. "RedundantSystem.syc": three sensor failure detection system
in AADL Inspector, observe how the input selection logic works

Example 8. "SystemModeling.syc": architectural design of an isolette with requirements traceability

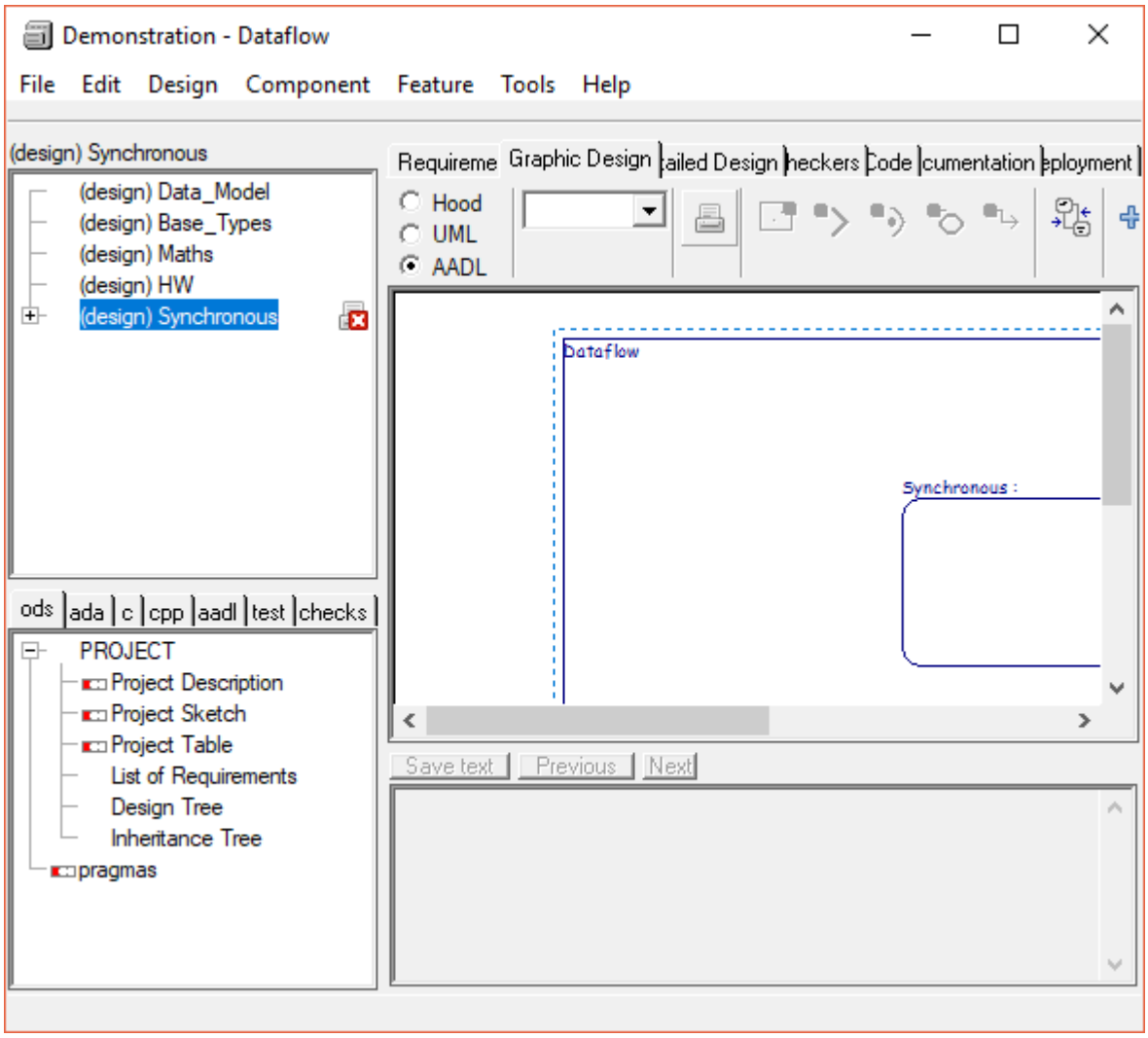
Example 9. "FlightSystem.syc": simplified aircraft flight system.

Example 10. "IMA.syc": another flight control system deployed with an Integrated Modular Architecture.

Example 11. "Pacemaker.syc": pacemaker control software.
in AADL Inspector, test scenarios can be used with the simulator.

Example 12. "Flows.syc": automatic generation of AADL flows and end to end flows.

After a `.syc` file has been loaded, you must select one of the listed **Designs** to be able to edit it. When selected, a **Design** is loaded in read-only mode by default. It is thus possible to navigate within the **Components** hierarchy without altering it. To be able to perform changes, the **Design** must be firstly locked (see details in next chapter).


Demonstration - Dataflow
File
Edit
Design
Component
Feature
Tools
Help
(design) Synchronous
(design) Data_Model
(design) Base_Types
(design) Maths
(design) HW
(design) Synchronous
Requireme
Graphic Design
ailed Design
heckers
ode
cumentation
ployment
Hood
UML
AADL
Dataflow
Synchronous :
ods
ada
c
cpp
aadl
test
checks
PROJECT
Project Description
Project Sketch
Project Table
List of Requirements
Design Tree
Inheritance Tree
pragmas
Save text
Previous
Next

5.3 Create a new Design

A **Project** is composed of several sharable **Designs** (your current application, reusable components, libraries, other sub-systems, ...). When opening **Stood** for the first time, only the predefined **Project** examples are available. To create a new **Project**, select the *New project...* option from the *File* menu and give it a name in the displayed dialog box:

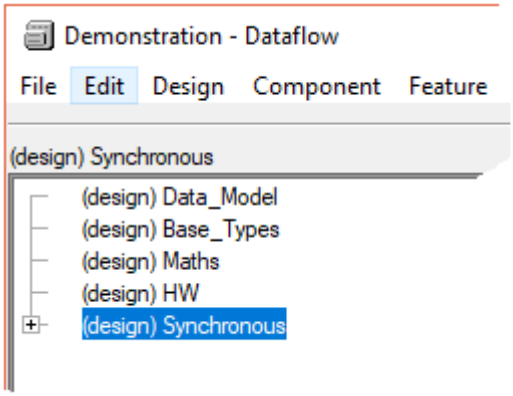
The created **Project**, however does not contain any data yet. To reference already existing **Designs**, select the *Add to project ...* option from the *File* menu, and select the chosen `Stood.sto` file from the file selector.

To create a new **Design**, select the *New design ...* option from the *Design* menu. Several options are available:

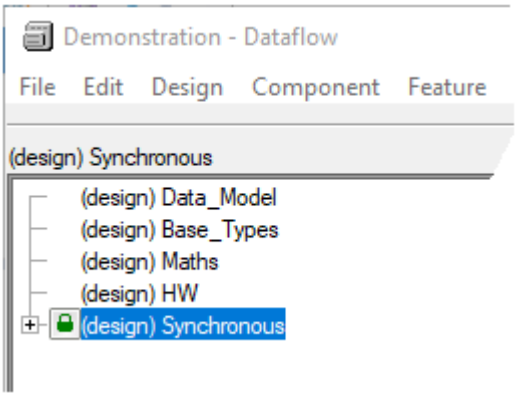
- **HOOD** designs: select *design*, *generic* or *virtual node*
- **AADL** designs: select *package*, *system*, *process* or *processor*

The corresponding entry will appear in the **Designs** list of the **Project**.

A single click on the name in the list, loads the **Design** in read-only mode. To load it in read-write mode, double-click the name or select the *Lock design* option from the *Design* menu. When loaded in read-write mode, a green padlock is shown near the **Design** name. This means that nobody else can open it in read-write mode.



loaded "*read only*"



loaded "*read write*"

The software design activities may now start. The following features are supported by **Stood 5.5**:

- import software requirements (text files or Reqtify 2018)
- reverse engineer legacy **AADL** models, or **Ada** or **C** source code
- perform architectural design activities graphically
- achieve complete detailed design and coding
- apply internal or remote model verification tools
- use code generators to producer **AADL**, **Ada**, **C** or **C++** source code
- generate design documentation in **HTML**, **PDF**, **RTF**, **ODT**,... .

When the **AADL** code generator is used, the embedded **AADL Inspector** companion tool may be automatically launched. **AADL Inspector** must be installed separately and provides a set of **AADL** processing tools such as **Cheddar** for schedulability analysis and **Marzhin** for dynamic run-time simulation. To associate the two tools, **AADL Inspector** install path must be specified in the `AI_PATH` variable of the customized `stood.ini` file (see chapter 1).

Please consult the provided documentation for further details:

- STOOD-Install Manual (Installation Guide)
- STOOD-AADL Manual (Stood for AADL User Manual)
- STOOD-AADL Tutorial (Building an AADL project step by step with Stood)
- AI User Manual (AADL Inspector User Manual)
- STOOD-Admin Manual (Advanced User Customization Guide)
- STOOD-Ada Coding Manual (Ada code generator options)
- STOOD-C Coding Manual (Guidelines for building C software with Stood)
- HOOD-Guidelines (Overview of the HOOD software design method)

In addition, other resources such as on line tutorials can be found on the support web site: http://www.ellidiss.fr

www.ellidiss.com

Sales office:
TNI Europe Limited
Triad House
Mountbatten Court
Worall Street
Congleton
Cheshire
CW12 1AG
UK
info@ellidiss.com
+44 1260 291 449

Technical support :
Ellidiss Technologies
24 quai de la douane
29200 Brest
Brittany
France

aadl@ellidiss.fr
+33 298 451 870