Linux Deployment Reference

Overview

This guide outlines the setup process for running FrameworX on Linux, tested on Ubuntu 22.04, Debian 12 (interface) and Debian 10.13 (no interface). FrameworX Runtime requires the cross-platform .NET Desktop Runtime: **.NET 8** for FrameworX 10.1.4 and earlier, **.NET 10** for FrameworX 10.1.5 and later. The setup involves installing the matching .NET runtime if it is not already available, importing necessary files, and configuring a connection between a Windows and Linux environment. It includes running web services on Linux, allowing Windows to connect via the platform, where additional features can be installed on Linux as needed. This approach facilitates a multi-project configuration, enabling seamless integration and functionality across both operating systems.

Overview
Steps to Install .NET on Linux (Ubuntu)
FrameworX Setup on Linux
Folder Layout on Linux
Remote Devices on a Linux Computer
Pythonnet for Python scripts

Steps to Install .NET on Linux (Ubuntu)

To install the cross-platform .NET runtime on a Linux machine, follow the steps below. Choose the version that matches your FrameworX release: **.NET 8** for FrameworX 10.1.4 and earlier, **.NET 10** for FrameworX 10.1.5 and later. The instructions are based on the Ubuntu distribution, but can be adapted for other distributions.

Update existing packages:

```
sudo apt update
sudo apt upgrade -y
```

Install the required packages:

```
sudo apt install -y wget apt-transport-https
```

Add the Microsoft Package Signing key:

```
wget https://packages.microsoft.com/config/ubuntu/$(lsb_release -rs)/packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
```

Notes on Version Handling

Ubuntu versions ending with zero:
The `lsb_release` portion of your `wget` command may fail when the Ubuntu version ends with a zero (e.g., `20.10`, `18.10`). This happens because `lsb_release` drops the trailing zero, resulting in an incorrect version string for the Microsoft package URL. To ensure the correct format (e.g., `20.10` instead of `20.1`), use the `printf` formatting shown below.

```
wget https://packages.microsoft.com/config/ubuntu/$(printf "%.2f" $(lsb_release -rs))/packages-
microsoft-prod.deb
```

Linux Mint and similar distributions:
For distributions such as Linux Mint, the Ubuntu base version cannot be obtained directly using `lsb_release`. The correct base version must instead be extracted from `/etc/upstream-release/lsb-release`, as demonstrated in the example below.

```
wget https://packages.microsoft.com/config/ubuntu/$(awk -F= '/DISTRIB_RELEASE/{print $2}' /etc
/upstream-release/lsb-release)/packages-microsoft-prod.deb
```

Update the repositories:

```
sudo apt update
```

Install the .NET SDK that matches your FrameworX version.

For FrameworX 10.1.4 and earlier — install .NET 8:

```
sudo apt install -y dotnet-sdk-8.0
```

For FrameworX 10.1.5 and later — install .NET 10:

```
sudo apt install -y dotnet-sdk-10.0
```

10.1.4 and 10.1.5 can be installed side-by-side on the same machine as isolated runtimes.

Verify the installation:

```
dotnet --version
```

FrameworX Setup on Linux

The UBUNTU version used for testing was 22.04. It was also tested with Debian 12. Other Linux distributions that support the matching .NET runtime should work as well, but there may be differences in how you install .NET.

Step 1

The cross-platform .NET runtime should already be installed on Linux — .NET 8 for FrameworX 10.1.4 and earlier, .NET 10 for FrameworX 10.1.5 and later. If not, go back to the section "Steps to Install .NET on Linux (Ubuntu)."

The Windows computer that runs the Designer also needs the matching .NET Desktop Runtime, even though the solution runs on Linux. Scripts in a Multi-Platform solution are compiled by the Designer, not on the Linux server. See Script Compilation Target.

Step 2

Download the TWebServices bootstrap from the FrameworX GitHub repository into the folder that will hold FrameworX on Linux, for example `/home/<user>/FrameworX`. This folder becomes the FrameworX root on Linux: TWebServices runs from it, and the product files are installed below it in Step 4.

```
git clone https://github.com/tatsoftdocs/FrameworX-v10.git ~/FrameworX
cd ~/FrameworX
```

If you don't have git installed, use the following command:

```
sudo apt install git
```

> **Do not unzip or copy the Windows installation onto Linux**
>
> Do not extract the Windows Bin zip on Linux, and do not copy the Windows installation folder or its `net10.0` subfolder to Linux. The Windows layout keeps the .NET 10 binaries in `fx-10\net10.0\`, next to the .NET Framework 4.8 binaries in `fx-10\`. On Linux there is no `net10.0` subfolder, so a copied Windows layout leaves the runtime files where FrameworX does not look for them. Remote device launch and the TWebServices file services then fail.
>
> Install on Linux only with the TWebServices bootstrap (Step 2), then **Install Product** / **Upgrade Product** from Solution Center on Windows (Step 4). For containers, use the images described in Container Deployment Reference.

Step 3

On Linux, run TWebServices so the Windows computer can connect. Inside the FrameworX root folder from Step 2, run the following command:

```
sudo dotnet TWebServices.dll
```

After you run this command, it should show you in the terminal that it is running and what is the IP and Port.

Step 4

After that, simply run FrameworX.exe on **Windows**, connect to Linux in “Server Information” > “Remote Server” > add the IP and port (it will be shown when you do the command in step 3), “Upgrade Product…” with the options you want to install on Linux (for example: HTML5, Designer/Remote, Protocol Drivers, etc.). Solution Center copies the product files from the Windows installation to Linux and writes them into the `fx-10` folder below the FrameworX root, in the Linux layout described in Folder Layout on Linux.

Step 5

Finally, click “Upload File…”, and upload the file you will use in the Linux environment. It is important to notice that the project has to be created with the multi-platform option.

Starting the solution without a Windows computer

Steps 4 and 5 use Solution Center on a Windows computer to install the product files and to upload the solution. After those steps, starting the solution needs no Windows computer. On Linux, run TStartup from the `fx-10` folder below the FrameworX root:

```
dotnet '/home/<user>/FrameworX/fx-10/TStartup.dll' /solution:'<solution folder>/MySolution.dbsln'
```

For the startup parameters, see Runtime Installation Reference.

The `frameworx` container image already holds the product files, so Steps 2 to 4 do not apply to it. See Container Deployment Reference and Docker Deployment Guide.

You create and build the solution in FrameworX Designer on Windows, with **Target Platform** set to Multiplatform.

Changed in FrameworX 10.1.6: Windows-target solutions are refused

From FrameworX 10.1.6, a runtime on Linux or in a container refuses to start a solution whose target platform is Windows (.NET Framework 4.8). The refusal applies to every start path: Solution Center, the Designer **Run Startup** button, **Actions Start Runtime**, the command line and the automatic start at boot. The message names the solution and the platform, for example:

```
The solution 'MySolution' targets Windows (.NET Framework 4.8) and cannot run on this computer (Linux).

Open the solution in Designer, change the target platform to Multiplatform in Solution Settings, run a
Build All, and try again.
```

To fix it, follow the message: in Designer, set **Target Platform** to **Multiplatform** in **Solution Settings**, run **Build All**, and upload the solution again. In 10.1.5, such a solution appears to start but never runs. Windows computers start every solution as before.

Also from 10.1.6, when the Multiplatform launcher stops during start-up, for example with no solution name, a port already in use, or a configuration file it cannot read, it exits instead of waiting indefinitely.

Folder Layout on Linux

Linux and macOS always run the .NET 10 runtime, and the folder layout differs from Windows:

Item	Linux / macOS	Windows
TWebServices	`<root>/TWebServices.dll`	`<install>\TWebServices.exe`
.NET 10 runtime binaries (TServer, TRemoteDevice and others)	Directly in `<root>/fx-10/`	`<install>\fx-10\net10.0\`
.NET Framework 4.8 binaries	Not present	`<install>\fx-10\`

For example, on Linux: `<root>/TWebServices.dll`, `<root>/fx-10/TServer.dll`, `<root>/fx-10/TRemoteDevice.dll`. There is no `net10.0` subfolder on Linux.

Remote Devices on a Linux Computer

A FrameworX server can start a Device module on another computer as a remote device. The server asks TWebServices on the receiving computer to start `TRemoteDevice` from that computer’s `fx-10` folder, and the remote device then connects back to the server.

Supported combinations

- Linux server to Linux receiver.
- Windows server to Windows receiver.
- Windows server to Linux receiver: supported from FrameworX 10.1.6. In earlier versions a Windows server cannot launch a remote device on a Linux receiver.

The receiver must be installed as described on this page, so that `TRemoteDevice` is in `<root>/fx-10/` and TWebServices runs from `<root>`.

Network ports

Port	Open on	Purpose
10108 (TCP)	Linux receiver, inbound	TWebServices: Solution Center connection, product install, file upload and remote device launch.
Runtime port of the solution (TCP)	Server, inbound	The remote device connects back to the server runtime. Defaults: 3101 for the Production profile, 3201 for Development.

Folders TWebServices serves

TWebServices file services only read, list and write files under these folders:

- The temporary folder of the computer.
- The FrameworX root folder, where TWebServices runs, including `fx-10` below it.
- The folders listed in `MachineSettings/SearchPaths.txt`. Folders you add to the solution folder list in Solution Center are stored in this file.

A request for any other path is refused, and so is a link that points outside these folders. TWebServices reads `SearchPaths.txt` when it starts, so restart TWebServices after changing it. Remote launch only starts `TRemoteDevice`, `TStartup` and `TRunModule` from the product folder.

Pythonnet for Python scripts

If Python scripts are to be used, pythonnet is essential. There are two options from this point: installing it globally or inside a virtual environment. Follow the steps below for both options:

Installing in a Virtual Environment:

Run the following command in the terminal to create the virtual environment (venv) in the current folder:

```
python3 -m venv ./venv_pythonnet
```

Once created, activate the virtual environment with the command below

```
source ./venv_pythonnet/bin/activate
```

After activation, upgrade pip to ensure compatibility and access to the latest packages:

```
pip install --upgrade pip
```

With the environment active and pip upgraded, install pythonnet:

```
pip install pythonnet
```

If you choose this option, you must run TWebServices.dll within the same virtual environment. Otherwise, FrameworX will not be able to locate the pythonnet module.

Installing Globally:

Check your python version

```
python3 --version
```

Based on your Python version, install the following packages to ensure that pythonnet can compile and run correctly:

```
sudo apt update && sudo apt install -y \
    python3-pip \
    python3.12-dev \
    build-essential \
    clang \
    libpython3.12 \
    libpython3.12-dev
```

Now install pythonnet using the following command.

```
 sudo pip3 install pythonnet --break-system-packages
```

Note: The --break-system-packages flag is required due to recent changes introduced by python.org. [More information can be found here.]

In this section...