



Knowledgebase > Setup / update > What to do when a TRACE32 screen driver library is missing?

---

## What to do when a TRACE32 screen driver library is missing?

2024-11-26 - Comments (0) - Setup / update

Since 26 June 2019 Lauterbach implemented a new screen driver model to support different GUI frameworks. The GUI specific software parts were split out into shared libraries (t32screen\*).

If one of the following shared object files

- t32screencde.so (for UNIX)
- t32screenqt4.so (for UNIX)
- t32screenqt5.so (for UNIX)
- t32screenwin.dll (for Windows)

or a related shared library is missing, TRACE32 PowerView displays an error message.

## Background information

Note

QT4 is not supported any more by the standard TRACE32 Release version. It is only supported by the [TRACE32 Software LTS Release 09/2020](#).

Customers use a wide range of Linux distributions, from very old to the latest versions. This creates compatibility challenges with the QT graphical user interface (GUI) framework used by TRACE32® software:

- Older Linux distributions do not support **QT version 5.x**.
- Newer Linux distributions no longer support **QT version 4.x**.

Since QT version 4.x and QT version 5.x are incompatible, additional considerations arise. Furthermore, some customers still prefer using the **MOTIF GUI** on Linux.

To address this, Lauterbach previously provided two executable variants for each target architecture on Linux:

- t32m\*: Supported the MOTIF GUI.
- t32m\* - qt: Supported the QT version 4.x GUI.

However, with the latest Linux distributions dropping support for QT version 4.x, introducing

a third executable for QT version 5.x (t32m\* - qt5) would have been impractical and increased installation size.

To streamline support for different GUI frameworks, Lauterbach implemented in 2019 a more efficient mechanism:

- GUI-specific components are now packaged as **shared libraries** (.so files on Linux, .dll files on Windows).
- When TRACE32® starts, it dynamically searches for these shared libraries and attempts to load them.

If no shared library is found, TRACE32® reports an error and aborts operation.

#### **Shared Library Loading Sequence on Linux:**

TRACE32® attempts to load the following shared libraries in order:

1. **t32screenqt5.so** - Supports QT version 5.x (only available for 64-bit Linux).
2. **t32screenqt4.so** - Supports QT version 4.x.
3. **t32screencde.so** - Supports the MOTIF GUI.

The software uses the first library it can successfully load.

This mechanism enables **auto-detection** of supported GUI frameworks:

- If QT version 5.x is not supported, t32screenqt5.so fails to load, and TRACE32® tries t32screenqt4.so.
- If QT version 4.x is also unsupported, TRACE32® tries t32screencde.so.
- If none of these libraries can be loaded, TRACE32® displays an error message and aborts.

#### **Shared Library Loading on Windows**

A similar mechanism is used on Windows. TRACE32® looks for the shared library:

- **t32screenwin.dll**

If this DLL is not found, TRACE32® aborts with an error message box.

## **Customizing GUI Library Loading in TRACE32**

Note

For regular TRACE32® installations, no additional steps are required. The following instructions are intended only for custom, handcrafted installations where enhanced flexibility is needed.

By default, TRACE32® searches for the required shared library (DLL/SO files) in the same

directory as the executable.

On **Linux**, TRACE32® will attempt to load GUI shared libraries in the following order:

1. t32screenqt5.so
2. t32screenqt4.so
3. t32screencde.so

You can override this behavior by setting an environment variable called T32SCREENS0.

- The variable defines a colon-separated (:) list of shared libraries (SO files) that TRACE32® should attempt to load in sequence.
- If TRACE32® cannot load any of the specified libraries, it will abort operation.

Examples for Linux

### Using a Relative Path

```
export T32SCREENS0=t32screencde.so:t32screenqt4.so
```

With this configuration:

1. TRACE32® will first try to load t32screencde.so from the directory where the executable resides.
2. If unsuccessful, it will try t32screenqt4.so from the same directory.
3. If both attempts fail, TRACE32® will terminate.

### Using an Absolute Path

```
export T32SCREENS0=/opt/t32/t32screenqt4.so
```

With this configuration:

1. TRACE32® will try to load /opt/t32/t32screenqt4.so (ignoring the executable's location).
2. If the file cannot be loaded, TRACE32® will terminate.

Example for Windows

### Using an Absolute Path

```
SET T32SCREENS0=C:\my_programs\t32dll\t32screenwin.dll
```

With this configuration:

1. TRACE32® will attempt to load C:\my\_programs\t32dll\t32screenwin.dll (ignoring the executable's location).
2. If the file cannot be loaded, TRACE32® will terminate.

# Troubleshooting

If TRACE32® is unable to load a DLL (Windows) or an .so file (Linux), it displays an error message clearly indicating which files it attempted to load.

- **For UNIX-like operating systems:**

This information is printed to the **standard error (stderr)**. To view the error message, you must start TRACE32® from a terminal or shell command line.

- **For successful starts:**

If TRACE32® starts without issues, you can use the following command in the TRACE32® command line to verify which shared library was loaded:

```
VERSION .SOFTWARE
```

This command opens a window displaying the details of the loaded DLL or .so file.

## Linux-Specific Hints

Missing System Shared Libraries

TRACE32® relies on system shared libraries to render its GUI on Linux. To use a specific GUI framework, ensure the required shared libraries are installed on your system.

To identify missing shared libraries, use the `ldd` command-line tool.

**Example 1:** Diagnosing a missing library for QT5

If `t32screenqt5.so` cannot be loaded, run:

```
ldd t32screenqt5.so
```

This will display diagnostic output indicating which libraries are missing.

**Example 2:** Diagnosing a missing library for QT4

If the QT4 system shared libraries are not installed, running:

```
ldd t32screenqt4.so
```

might yield output like this:

```
linux-vdso.so.1 (0x00007ffe52d2e000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0
(0x00007f47cd768000)
libQtGui.so.4 => not found
...
```

### Finding the Missing Package

Once you identify the missing library, determine the package name that provides it. Below are common libraries and their associated GUI frameworks:

- **MOTIF:** `libXm.so.4`
- **QT4:** `libQtGui.so.4`

- **QT5:** libQt5Widgets.so.5

The method to find the package depends on your Linux distribution:

- **APT-based distributions (Debian, Ubuntu, Linux Mint, etc.):**

Install apt-file if not already installed:

```
sudo apt install apt-file
apt-file find <library>
```

- **Zypper-based distributions (SUSE Linux, etc.):**

```
zypper search -f <library>
```

- **YUM-based distributions (Older Red Hat, Fedora, etc.):**

```
yum whatprovides '*/<library>'
```

- **DNF-based distributions (Recent Red Hat, Fedora, etc.):**

```
dnf provides <library>
```

- **Pacman-based distributions (Arch Linux, Manjaro, etc.):**

```
pacman -Fs <library>
```

- **Gentoo:**

Search for the package online.

## MOTIF-Related Considerations

- Previously, Linux provided two executables per architecture:
  - t32m\* (MOTIF GUI)
  - t32m\*-qt (QT GUI)
- With the new shared library mechanism, t32m\*-qt has been replaced by a small bash script. If you update TRACE32® by copying only the t32m\*-qt file (without copying t32m\*), the script will invoke the old software version using the MOTIF GUI.

**Hint:** Always copy both t32m\* (without -qt) and t32m\*-qt during updates to avoid this issue.

To enforce the use of the MOTIF GUI, set the following environment variable:

```
export T32SCREENS0=t32screencde.so
```

To make this permanent, add the line above to your ~/.bashrc file.

## Deprecated Executable Warning

If you encounter a warning like:

Warning: starting `/home/testuser/t32/bin/pc_linux64/t32marm-qt` is deprecated, see <https://www.lauterbach.com/3737>

update your start script or command line to replace `t32m*-qt` with `t32m*` (without `-qt`).