LAUTERBACH
DEVELOPMENT TOOLS

Floating Licenses

MANUAL

Floating Licenses

Floating Licenses

Version 06-Jun-2024

History

02-May-2024 New chapter “License Roaming”.

02-Oct-2023 Reprise links in chapter “License Management Server” updated.

05-Sep-2023 The general conditions for the TRACE32 Simulator License described in more detail in chapter 'TRACE32 Simulator License as Base License'.

22-Sep-2015 Initial version of the manual.

Introduction

In addition to its hardware-based tools Lauterbach also provides so-called software-only tools. In order to use a software-only tool one or more floating licenses are required.

In this document, we will introduce these licenses that are served by a floating license server to one or multiple license clients.

TRACE32 Floating License Overview

TRACE32 Front-End License as Base License

A **TRACE32 Front-End License** licenses the use of the TRACE32 PowerView GUI and its high-level debug features. It is provided as base license for various use cases. Here a few examples:

- Debugging a Virtual Target.
- Debugging a real target by communicating via TCP/IP with a gdb agent.
- Debugging a real target via the standard USB interface using the USB stack of the OS.
- Debugging a core simulation by using a GTL library.

A TRACE32 Front-End License covers a processor architecture e.g. ARM/Cortex, PowerPC or Intel® x86/x64.

The TRACE32 Front-End License is bound to a software warranty contract.

The following other floating licenses can be used together with a TRACE32 Front-End License:

- **TRACE32 Back-End License**

  If the low-level debug driver, the so-called HostMCI, is provided by Lauterbach, a TRACE32 Back-End License is needed. A TRACE32 Back-End license covers a sub-architecture and a communication channel. E.g. *USB x86/x64 Debug Back-End* or *GTL Cortex-M Debug Back-End.*

- **TRACE32 Multicore License**

  If more than one TRACE32 PowerView GUI has to be started to debug the target under test (AMP system), a TRACE32 Mulitcore License is required (see **Floating License Pools**).

- **TRACE32 Trace License**

  The decoding of core trace information for trace analysis and display requires a TRACE32 Trace License, e.g. if core trace information such as Intel® Processor Trace is provided via a standard USB interface using the USB stack of the OS.

- **TRACE32 Integration License**

  Performing a Simulink® PIL Simulation requires a TRACE32 PIL Simulation License. All other 3rd-party tool integrations with TRACE32 are currently free of charge.

TRACE32 Simulator License as Base License

The **TRACE32 Instruction Set Simulator** supports a number of functions license-free. However, if it is used for test automation, a simulator license is required as soon as 50 script commands/API operations are executed after the first "single step" or "go".

This affects the following test scenarios:

- Larger PRACTICE scripts running on the TRACE32 Instruction Set Simulator.
- Controlling the TRACE32 Instruction Set Simulator via the TRACE32 Remote API.
- Controlling the TRACE32 Instruction Set Simulator via a third-party tool integration.
- Controlling the TRACE32 Instruction Set Simulator as a TCF agent.
- Controlling the TRACE32 Instruction Set Simulator via the TRACE32 GDB API.

A *TRACE32 Simulator License* covers a processor architecture e.g. ARM/Cortex, PowerPC or Intel® x86/x64.

The TRACE32 standard software warranty applies. Once this has expired, the TRACE32 Simulator License is bound to a software warranty contract. If the hardware-based TRACE32 debugger is connected to the hosts computer, its license can be used.

PBI=*SIM USB	start a TRACE32 Instruction Set Simulator which is licensed using a hardware-based TRACE32 debugger connected to the host computer via USB.
PBI=*SIM NET NODE=<*node_name*>	start a TRACE32 Instruction Set Simulator which is licensed using a hardware-based TRACE32 debugger connected to the host computer via Ethernet.

The following other floating licenses can be used together with a TRACE32 Simulator License:

- **TRACE32 Trace License**

  If more the 64K records of core trace information are loaded with the **LA.IMPORT** command for decoding a TRACE32 Trace License is needed, e.g. *Trace License for Intel® Processor Trace*.

- **TRACE32 Integration License**

  Performing a Simulink® PIL Simulation requires a TRACE32 PIL Simulation License. All other 3rd-party tool integrations with TRACE32 are currently free of charge.

Licensing Terms Glossary

Terms for Floating Licenses with RLM

Client License File **client.lic**	Tells the License Client the RLM Server name and port number. **Typically this file is only one line of text**: `HOST` <RLM Server hostname> `PORT` <port>
ISV **(Software Publisher)**	Independent Software Vendor: a company who uses RLM for licensing their products. Lauterbach is an ISV.
ISV Service **Lauterbach** **ISV Service**	Another running instance (fork) of the RLM Server. It validates the Lauterbach License File with the keys from a Lauterbach Certificate File, and serves floating licenses to License Clients.
Lauterbach Certificate File **lauterbach.set**	This file contains the Lauterbach Public Key to validate a Lauterbach License File. It is used by the ISV Service.
Lauterbach License File **lauterbach-xxxxxxxxxxxx.lic**	**One** file, includes all floating licenses issued by Lauterbach for the same RLM Host ID. **You need to modify this file with your server name and the ports you want to use for RLM Server and ISV Service:** `HOST` <RLM Server hostname> <RLM HostID> <port> `ISV lauterbach port=`<ISV Service port>
License Server	Your license server machine with the Lauterbach License File, running an RLM Server and the Lauterbach ISV Service.
License Client	The code built into TRACE32 to check out floating licenses.
RLM Host ID	Unique identification number of your License Server machine. Your licenses will be “node-locked” to this ID.
RLM Server	The **Reprise License Management** (RLM) **Server**, managing one or more ISV servers (which hand out floating licenses).

How To Install Floating Licenses - Overview

1. Download and install the **RLM License Administration Bundle** for your License Server machine. For the download location, see **"License Management Server"**, page 9.
2. Use it to look up your RLM Host ID for your License Server machine
3. Send us the **RLM Host ID**, so we can generate the **Lauterbach License File** for your server**.**
4. Download and unpack the Lauterbach Certificate **lauterbach.set** and copy it into the RLM Server installation directory (see **"Lauterbach Certificate Details"**, page 9).
5. When you receive your **Lauterbach License File**, copy it into the RLM Server installation directory.
6. **Modify the first lines of your Lauterbach License File**: set your License Server machine **hostname** and **port** .
7. Configure your License Server to start the RLM Server Executable (rlm or rlm.exe) at boot time.
8. **Set the RLM_LICENSE environment variable** or make a client license file, to tell the license clients the RLM License Server name (or IP address) and port number.

NOTE: **RLM is not FLEX*lm***

- Matt Christiano started FLEX*lm* development in 1988. In 2005, he left Macrovision and founded Reprise Software, Inc. to develop RLM.
- **A FLEX*lm* HostID and an RLM Host ID are not the same thing!** If a license file does not work because you submitted a FLEXlm hostid, you will have to order and pay for a license transfer to get a new one.
- lmutil and rlmutil are different tools, from different companies, and speak slightly different protocols - so please never try to get an "lmstat" from an RLM server and an "rlmstat" from a FLEX*lm* server.

How To Upgrade Floating Licenses

1. **Back up** the existing Lauterbach License File from your RLM Server into a different directory or onto a remote PC/workstation.
2. **Modify your new Lauterbach License File**: set hostname and port of your License Server machine (just copy the hostname/port data from the old license file).
3. **Copy** the - now modified - new Lauterbach License File into your RLM Server directory. License Files generated by Lauterbach always contain all valid licenses for your RLM server, as identified by its rlmhostid, so there should only be exactly one active License File at all times.
4. **Re-read the licenses from the new License File** to make them available to your clients. You can do this via the RLM Server web interface, or with rlmutil.

License Management Server

Downloads

Server Download	**https://reprisesoftware.com/support/admin/license-administration-bundle/** Download the **RLM License Administration Bundle** for your server. Contains RLM Server, documentation and a performance test.
Certificate Download	**https://www.lauterbach.com/faq/lauterbach.set** This contains the **lauterbach.set** file you need.
Lauterbach License file	**lauterbach-<rlmhostid>.lic** You will receive this file via eMail.

License Management System Details

LM Vendor	Reprise Software, Inc. - **https://reprisesoftware.com/products/reprise-license-manager-rlm/**
LM Version (minimum)	RLM v10.1BL2
Server platforms for ISV Lauterbach	Windows 32/64-bit, Linux, 32/64-bit, MacOS. For exact versions, please see the download page above.

Lauterbach Certificate Details

Supported RLM server versions	RLM Server v9.0 and later (on supported server platforms). Please note that Lauterbach Certificates prior to RLMv9 were not enabled for running an RLM Server on Linux/64bit.

The same certificate file can be used with multiple RLM versions and RLM Server platforms.

Lauterbach Daemons (for RLM Servers v4.0BL4 and v5.0)

These daemon executables were replaced by the Lauterbach Certificate (see above).
If you still have a 'lauterbach' daemon executable, please delete it !

The certificate system makes sure RLM Server and ISV Service versions are always identical.

RLM Server and Client use public key cryptographic signatures. When the RLM Server starts up, it looks for valid license files. For each one it finds, it checks if already an ISV service is running. If not, it starts another instance of itself that acts as this ISV Service. The ISV service then reads the ISV's license file, and also the ISV's public key from the 'ISV settings file'. Using the public key, it verifies the licenses found in the license file(s). If the ISV service receives a license request from an RLM client for a given product name and version, it checks if the maximum license count for this product name/version still allows another checkout. If a checkout is possible, the iSV service grants a license to the RLM client.

Lauterbach licenses are special in one respect: we issue permanent licenses. This and the upgrade mechanism we use makes it necessary to use the RLM Token System.

The maximum possible TRACE32 revision that can be run with a given maintenance license is identified by a version string of the form YYYY.MM, e.g. 2015.03 for March, 2015. The maximum version is tied to a maintenance contract with a given serial number. The RLM license for this serial number must have the serial number as part of the product name, so it can be replaced with a new one. But the RLM Client cannot know which serial number it will get. With the RLM Token System a generic name like 't32.frontend.arm' can be used as a token that is aliased to one or more primary licenses with the proper serial number in the product name.

Here is an example RLM license file:

```
# HOST [rlmserver] 0090f1f2f312 [port]
HOST localhost 0090f1f2f312 5055
ISV lauterbach lauterbach port=5056
LICENSE lauterbach t32.frontend.arm 2016.03 31-mar-2016 token_locked
    token="<t32.115020000979.maint 2016.03 1>" sig="60P0451698N6SCU48H
    ACF9N4WUGA4BYM95TTED022HF4S1WBFKQGRC44JVP4QV5DS0GE6XCG0R"
LICENSE lauterbach t32.115020000979.maint 2016.03 31-mar-2016 2
    issued=26-feb-2015 replace sig="60P045212CWC4CSQFA
    FKNR6K9Y016JE6VBTESE822G2YHBS0D2K504GR9AWHPW28S6EE0EKFWC"
```

The RLM Host ID is `0090f1f2f312`, the RLM master server runs on port 5055, ISV service 'lauterbach' on port 5056 - if your users are on a WAN, the firewall admins need to make sure they can reach both ports for successful license checkout.
The file contains one active floating license for `t32.frontend.arm`, a token pointing to `t32.115020000979.maint`, version `2016.03`.
This license is a test license, it is not `permanent`, but will expire on `31-mar-2016`, and it enables `2` concurrent users to work with TRACE32 for ARM in a FRONT-END setup.

License Server Setup

If you are an experienced License Administrator with administrator privileges for one or more company-internal license servers, you can probably skip this section.

RLM is not difficult to install. If you can download a file from the internet, unpack it, copy it to a suitable server directory, and start it, you can get a server up and running in less than one hour.

If you have not installed any license system before, maybe calculate two to three hours, plus reading time for the RLM manual. The added material in this section might be useful, too.

How to get the RLM Host ID

First, download the current RLM License Administration Bundle and install it on your License Server machine. Then, alternatively, get the RLM Host ID with either...

...the rlmutil command line utility:

- On a command line, invoke 'rlmutil rlmhostid ether'.
- This will print the 12-digit (48bit) RLM Host ID for your machine.
- If multiple RLM Host IDs are reported, please select one to lock your licenses to.

...the RLM web server interface:

- Start the RLM Server executable.
- Use a web browser to access the RLM License Server Administration page (port 5054 by default)
- Click on "System Info", and read the RLM Host ID (the 12-digit 'Ethernet' one, please).
- If multiple RLM Host IDs are reported, please select one to lock your licenses to.

How to Configure an RLM Server for Auto-Start

For Windows, the RLM server has a special command line option to install it as a service.
You can also write a batch file to invoke it (with any options you like) in the correct subdirectory, and add a link to it in the All Programs\Autostart folder.

For Linux, here is an RLM autostart example for the rc.d system (e.g. used by SLES).

(1) Copy the script file rlm-rc.txt to /etc/rc.d/rlm

```
sudo cp /media/MEMORY_STICK/rlm-rc.txt /etc/rc.d/rlm
sudo chmod 755 /etc/rc.d/rlm
```

(2) Link to activate it in runlevels 3 and 5:

```
sudo ln -s /etc/rc.d/rlm /etc/rc.d/rc3.d/S07rlm
sudo ln -s /etc/rc.d/rlm /etc/rc.d/rc5.d/S07rlm
```

(3) Manual start / stop / restart, e.g. first manual start after installation:

```
sudo /etc/rc.d/rlm start
```

RLM Autostart Script rlm-rc.txt

Here is the first part of the script:

```
#! /bin/sh
# rlm           Start/Stop rlm
### BEGIN INIT INFO
# Provides: rlm
# Required-Start: $network $remote_fs
# Required-Stop: $network $remote_fs
# Default-Start: 3 5
# Default-Stop: 0 1 2 6
# Description: Start the rlm daemon
### END INIT INFO

RLM_BASEDIR=/opt/rlm
RLM_BIN=$RLM_BASEDIR/rlm
RLM_LOG=$RLM_BASEDIR/log/rlm.log
RLM_OPT="-ws 5054 "
RLM_UTI="$RLM_BASEDIR/rlmutil"
RLM_USER=hpatzke
# option syntax          : comment
# -ws <port>             : set web server port (default:5054)
# -nows                  : disable web server
# -c <license_file>      : license file name to use
# -dat                   : change license file xtension from .lic to .dat
# -x [rlmdown|rlmremove] : exclude [name] command processing
# -q                     : stop (quit) server
# -dlog +/pathname       : set debug log to pathname, "+" appends data
#                          not used, b/c ISV servers without option file
#                          will log to stdout by default

test -x $RLM_BIN || exit 5
```

Here is the second part:

```
. /etc/rc.status

# Shell functions sourced from /etc/rc.status:
#      rc_check         check and set local and overall rc status
#      rc_status        check and set local and overall rc status
#      rc_status -v     ditto but be verbose in local rc status
#      rc_status -v -r  ditto and clear the local rc status
#      rc_failed        set local and overall rc status to failed
#      rc_reset         clear local rc status (overall remains)
#      rc_exit          exit appropriate to overall rc status

# First reset status of this service
rc_reset

case "$1" in
    start)
        echo -n "Starting RLM daemon "
        echo "---start server---" >>$RLM_LOG
        su - $RLM_USER -c "$RLM_BIN $RLM_OPT >>$RLM_LOG &"
        rc_status -v
        ;;
    stop)
        echo -n "Shutting down RLM daemon "
        echo "---stop server---" >>$RLM_LOG
        su - $RLM_USER -c "$RLM_UTI rlmdown -q lauterbach "
        su - $RLM_USER -c "$RLM_UTI rlmdown -q "
        sleep 2
        killproc $RLM_BIN
        rc_status -v
        ;;
    restart)
        ## Stop the service and regardless of whether it was
        ## running or not, start it again.
        $0 stop
        $0 start
        # Remember status and be quiet
        rc_status
        ;;
    *)
        echo "Usage: $0 {start|stop|restart}"
        exit 1
        ;;
esac
rc_exit
```

Floating License Transfer

Explanation for Managers

Please order “LA-8029 TRANSFER-FLOATING”, including the RLM Host IDs of the old and the new server, plus the list of licenses (serial numbers) the transfer applies to.

Explanation for Engineers

Organizations change, work moves from one department to another, sometimes to a different continent. Or just a server machine is replaced by a new one. In any case, this is a likely question you will ask:

“Can you modify our license files, so they will reside on the correct servers?”

Sorry, no, we can’t -- because of the way how Lauterbach Floating Licenses work:

1. Floating licenses need to be locked to the license server.
2. The identification we use with RLM is the "rlmhostid". The "rlmhostid" we use is derived from the network interface MAC address of the server.
3. Our floating licenses are **permanent** - they don't need to be renewed annually. (The only exception from this general rule are temporary licenses for testing, these expire at the end of a few months.)

If you want to use the newest TRACE32 version, you might need to update your Front-End licenses. Back-End and Trace licenses don’t need to be updated at all.

Here is what we at Lauterbach have to do for a "server change":

- make new licenses with new serial numbers for the new server, and then
- blacklist the old serial numbers in our next software release.

Therefore we need some legal commitment that the old licenses are not used anymore in the future. Having customers sign an extra legal statement for this does not work (we tried this), hence we created a stock unit that needs to be ordered. The T&C for this force the old licenses to be taken out of use.

“OK -- but how do we actually do it?”

Please order “LA-8029 TRANSFER-FLOATING”, including the RLM Host IDs of the old and the new server, plus a list of serial numbers the transfer applies to. You will receive a new license file for the new server.

If you are not sure the new license server layout is fixed yet, please ask for a bunch of temporary licenses for testing first, before any permanent licenses are generated - temporary licenses will expire, so all is well. But any transfer of permanent licenses does need an LA-8020 TRANSFER-FLOATING order.
(Sorry, we can make no exceptions to this rule.)

License Client Setup

NOTE:	Floating License Servers and Clients are normally set up by the License Administrator of your organization. TRACE32 end-users should normally not need this description.

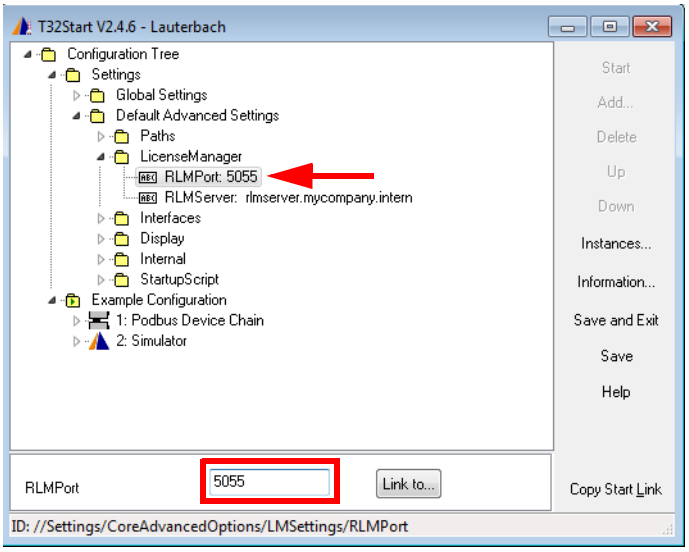
If the TRACE32 configuration file selects a software interface (such as GDB, GDI, CADI, MCD, etc.) for debugging, TRACE32 will automatically (1) check for a valid license in a USB dongle or (2) check out a floating license via Reprise Software's License Manager (RLM).

- TRACE32 has 'built-in' the RLM Floating License Client for Windows, Linux and MacOS X

The Floating License Client (RLM Client) needs to know which (RLM) Server to contact to get the license. We present three alternatives to do this, please use only one of them! Each one will tell TRACE32 to get the license from a server with the name `rlmsrv1.mycompany.intern` and port number `5055`.

Alternative A: Enter Port and Server in t32start.exe

1. Navigate to the folder item **RLMPort**.
2. Enter the port number in the box at the bottom of the window.



Alternative B: Create the environment variable RLM_LICENSE

RLM_LICENSE=*<port>@<rlmserver>*

```
//To list all variables, run set on Windows (cmd) or printenv on Linux
RLM_LICENSE=5055@rlmsrv1.mycompany.intern
```

Alternative C: Create a 'client license file' CLIENT.LIC

1. Any name is ok, but the file name extension must be **.lic** (capital or lower case letters are OK).
2. Place this in the working directory that is active when TRACE32 is started.

The CLIENT.LIC file consists only of one line of text. This tells the client where to find the server:

```
HOST rlmsrv1.mycompany.intern PORT 5055
```

HOST *<rlmserver> <something> <port>*

<rlmserver>	Host name or IP address of your RLM Floating License Server
<something>	On the SERVER, this contains the RLM HostID, but on a CLIENT anything that does not contain spaces is fine - e.g. the string "PORT" looks nice.
<port>	Port number of your RLM Floating License Server

The syntax for this option, available from TRACE32 build rev. 56078, is identical to (A). Example:.

```
;TRACE32 Test Configuration cfg-lictest.t32
OS=
ID=T32LT
SYS=C:\T32
TEMP=C:\T32\TEMP
HELP=C:\T32\PDF

; This configuration block tells TRACE32
; how to contact the RLM server.
LICENSE=
RLM_LICENSE=5055@rlmsrv1.mycompany.intern

;Connection to Host
PBI=MCILIB

;Screen Settings
SCREEN=
HEADER=TRACE32 License Test

;Printer Settings:
PRINTER=WINDOWS
```

The optional parameter `TIMEOUT=<time>` allows to automatically return checked out but unused licenses. The value of this parameter holds the minutes of inactivity before licenses are returned. A minimum of 15 minutes is defined. Setting the optional parameter to zero (`TIMEOUT=0`) is the same as no such line and will result in never automatically returning unused floating licenses. Inactivity in the context of this parameter is time spent in down state.

Example: In this example, the option `TIMEOUT` will result in returning licenses after 60 minutes after switching to **SYStem.Mode Down**.

```
LICENSE=
RLM_LICENSE=5055@rlmsrv1.mycompany.intern
TIMEOUT=60
```

Just in case you get TWO or THREE port numbers from your (fellow) administrator(s):

On the License Server machine, at least two server processes must be running to serve licenses. By default these will use at least three ports:

- **<port>: the RLM Server port**

  The RLM Server tells the RLM Client(s) on this port, where to look for the ISV Service.

  => Your CLIENT.LIC file contains only this RLM Server port number!

- **<isv port>: the ISV Service port**

  The ISV Service does the actual license handling. If you don't set this port number (in the server license file), the RLM Server will set <isv port> to a random free port number.

  If you have RLM-licensed products from several Independent Software Vendors (ISVs), your server will run multiple ISV Service processes - one for each ISV.

  The ISV Service is actually a 'copy' (fork) of the RLM Server process, but it is configured to act as the Lauterbach ISV Service. It uses a **lauterbach.set** file to validate the license file.

- **<web_port>: the RLM Server's built-in web server port**

  RLM Server comes with a built-in web server. Via this web (HTTP) server port, anyone can check the current license status with a web browser.

  To see the license status, just go to the website: `http://<rlmserver>:<web_port>/`

  (no spaces), then click on the vendor and product names to get to their license status pages.

  Example: `http://rlmsrv1.mycompany.intern:5054/`

If your RLM Server is behind a firewall, ask your firewall administrator can unblock these ports. RLM Server and ISV Service ports must both be accessible for the license clients, otherwise license checkout will fail.

Multiple RLM servers

If you have more than one RLM server, you can either use multiple CLIENT.LIC files, or add the second server to the RLM_LICENSE setting. Here is an example for Windows:

```
RLM_LICENSE=5055@rlmserv1.company.intern;5055@rlmserv2.company.intern
```

For Linux/Unix, you have to use a colon as the path separator (instead of a semicolon):

```
RLM_LICENSE=5055@rlmserv1.company.intern:5055@rlmserv2.company.intern
```

Supported Operating Systems

Floating licenses can currently be used with TRACE32 versions compiled for these operating systems:

License Client	
Windows 32-bit	Included in TRACE32/PowerView GUI application (builds earlier than rev. 31930 use t32lm.dll)
Windows 64-bit	Included in TRACE32/PowerView GUI application
Linux 32-bit	Included in TRACE32/PowerView GUI application
Linux 64-bit	Included in TRACE32/PowerView GUI application (from build rev. 26801)
MacOS X 64-bit	Included in TRACE32/PowerView GUI application (from build rev. 69145)

Floating License Pools

If the TRACE32 configuration file selects a software interface (such as GDB, GDI, CADI, MCD, etc.) for debugging, with no Lauterbach TRACE32 hardware involved, we call this a 'software-only' configuration.

For these, TRACE32 checks for a valid maintenance license (1) in a USB dongle or (2) checks out a FRONTEND floating license via Reprise Software's License Manager (RLM).

If you have a software-only setup that additionally requires a BACKEND license, TRACE32 does also a checkout for a BACKEND license via Reprise Software's License Manager (RLM).

If any of the required licenses is missing, TRACE32 will operate in demo mode, the same behavior as if a license was missing in a debug cable.

There is an essential difference between 'software-only' and 'cable-based' licenses:

- If you debug a real target, e.g. an SoC, via a TRACE32 device and its attached debug cable, your 'target system' is well-defined - it consists of anything connected to the debug cable.
- If you debug a virtual target, with TRACE32 in a FRONTEND setup, there is no reliable way to define what belongs to your 'target system' and what does not.

If you debug an Asymmetric Multi Processing (AMP) system with a TRACE32 device, you need only two license entries in your cable:

- One core-family license plus one multicore license, or - alternatively -
- Two different core-family licenses.

All started TRACE32/PowerView GUIs will 'share' the licenses in the single debug cable.

If you debug an AMP system with TRACE32 in FRONTEND mode, e.g. with a virtual target, each started TRACE32/PowerView GUI normally needs its own FRONTEND license.
This means you need (many) more licenses than with a real target and a debug cable.

As a service to our good long-term customers, who debug on virtual platforms and real targets alike, we introduced the **License Pool** feature for sharing 'software-only' licenses.

For FRONTEND setups, the MULTICORE floating license was created. For AMP debugging, this enables multiple TRACE32 instances to share the same FRONTEND license in a License Pool.

The License Pool feature is also used for BACKEND and SIMULATOR licenses. The only difference to FRONTEND setups is that you don't need an extra MULTICORE license to enable pooling.

Software-Only License Types

TRACE32 can pool these software-only license types:

- **FRONTEND** license (e.g. t32.frontend.arm)
  A FRONTEND license is checked out when the TRACE32 PowerView GUI starts in software-only mode:
  - either from RLM floating license management server (default)
  - or from TRACE32 License Pool Server (if configured)
- **BACKEND** license (e.g. t32.cortexm.gtl)
  A BACKEND license is checked out when the first connection to any back-end is activated, e.g. by the commands **SYStem.Mode Attach**, **SYStem.Mode Up**, or by calling a RemoteAPI function.

  BACKEND requests are automatically granted by the Pool Server, if the same license has been successfully checked out before. Otherwise the Pool Clients check out from the Pool Server, the Pool Server in turn performs a checkout from the RLM Floating License Manager.
- **MULTICORE** license, t32.multicore.all
  To enable FRONTEND license pooling, we introduced this license type. A MULTICORE license is checked out by Pool Server, when the first Pool Client requests a FRONTEND license.
- **SIMULATOR** license (e.g. t32.simulator.tricore)
  A SIMULATOR license is checked out when the first instruction set simulator is activated, e.g. by the commands **Step**, or **Go**.
- SIMULATOR requests are automatically granted by the Pool Server, if the same license has been successfully checked out before. Otherwise the Pool Clients check out from the Pool Server, the Pool Server in turn performs a checkout from the RLM Floating License Manager.

To configure a License Pool:

1. Open the TRACE32 configuration file (default config.t32).
2. Add a `LICENSE=` configuration block (if not already present), then to this block.
3. Add an entry `POOLPORT=<number>`. The *<number>* sets the pool's TCP/IP port.

To illustrate the result of these steps, here is an example:.

```
;TRACE32 Test Configuration cfg-x1.t32
OS=
ID=T32X1
SYS=C:\T32
TEMP=C:\T32\TEMP
HELP=C:\T32\PDF

; This configuration block activates TRACE32 license pooling.
; The first TRACE32 PowerView GUI started with a POOLPORT becomes
; "server".
; Subsequent instances are clients and will connect to the server.
; Pool Status is available via SETUP.DRIVER and LICENSE.List.
; FRONTEND pooling will try to check out a MULTICORE license.
; If this fails, a second identical FRONTEND license checkout
; (e.g. t32.frontend.arm) will be tried.
; BACKEND pooling does not require an extra multicore license.
; Communication protocol is TCP/IP. Make sure the chosen port number
; is not in use by anything else in your IT infrastructure.
LICENSE=
POOLPORT=5655

;Connection to Host
PBI=MCILIB

;Screen Settings
SCREEN=
HEADER=TRACE32 License Pool Test [X1]

;Printer Settings:
PRINTER=WINDOWS
```

NOTE: You will likely need different path names and PBI settings.
Please adapt this example as needed.

For easy access to an index of available options and their documentation, please enter this at the TRACE32 command line:
`HELP.Find "PBI="`

Please note that you can currently not enter the POOLPORT entry in t32start.exe.

The first TRACE32 instance that can successfully bind and listen to this local TCP/IP port becomes the **Pool Server**. TRACE32 instances that cannot bind and listen to the port become a **Pool Client**.

1. When it starts, the Pool Server binary first makes a FRONTEND license checkout for itself (from the RLM Floating License Server).
2. When a Pool Client starts, it connects to the Pool Server.
3. After successful connect, the Pool Client requests a FRONTEND license from the Pool Server.
4. The Pool Server checks out a MULTICORE license from the RLM Floating License Server.
5. If the Pool Server holds one FRONTEND and one MULTICORE license (or - as a fallback - two FRONTEND licenses), all subsequent Pool Client FRONTEND requests are immediately granted by the Pool Server, up to the maximum permissible pool size.

Please note:

- If a MULTICORE license checkout is not successful, the Pool Server tries to check out a second FRONTEND license (identical in type to the first one it already has).

  If this attempt also fails, the Pool Server denies the FRONTEND license to the Pool Client, and the Pool Client (TRACE32 instance) will terminate with an error message.

- The Pool Server will only allow connections if the License Pool can hold an additional client. If not, the Pool Client terminates with an error message (Pool Full).

  Currently, a maximum of eight TRACE32 instances can participate in one pool (i.e. one pool server, and seven pool clients).

  If you need more than eight instances for the same core family, you will have to configure a new pool (i.e. assign a different port number in the configuration file of TRACE32 instances #9..16).

- All TRACE32 instances in a pool must be for the same CPU family (e.g. ARM, PPC, MIPS4K).

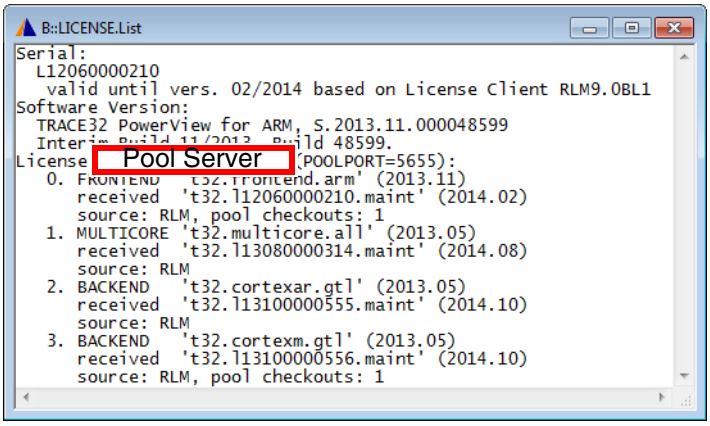
  "Mixed operations", e.g. 5 x ARM and 3 x PPC within the same pool - is not possible. (For such a setup you need two pools, one for PPC and one of ARM.)

- All pool instances should be started from the same binary executable image on disk. The License Pool communication protocol will change over time. Using the same binary executable ensures that Pool Server and Pool Client can always communicate properly.

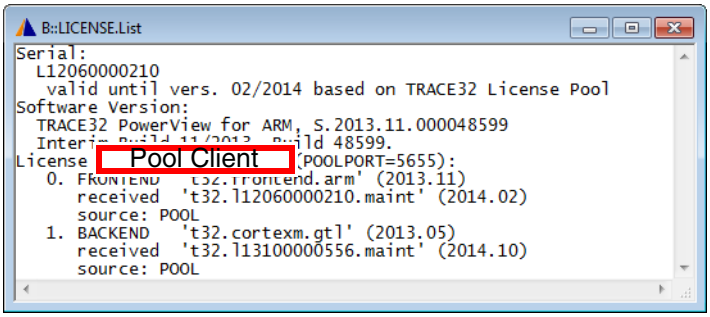
- License pools can only be set up within one Host PC (only a port number is required).

- The pool communication protocol is TCP/IP. Please be aware that you might have to explicitly allow this communication, e.g. in the Windows Firewall.

This is how the **LICENSE.List** window looks for a Pool Server:



This is how the **LICENSE.List** window looks for a Pool Client:



Please read the chronology below, if you want to know how this display was created.

Here is a chronology of the actions that resulted in the two screens above:

1. The user started the first TRACE32 instance. This instance could bind the POOLPORT, and so became Pool Server.

   This instance then automatically requested the 't32.frontend.arm' FRONTEND license from the RLM license server. The FRONTEND license is shown in "slot 0" of the Pool Server's **LICENSE.List**.

2. Then the user started the second TRACE32 instance. The POOLPORT was already in use, so this instance became Pool Client, and it contacted the Pool Server to get a FRONTEND license.

   This request from the Pool Client prompted the Pool Server to get a MULTICORE license. MULTICORE checkout was successful, and you can see the entry in Pool Server's "slot 1".

   Then the Pool Server forwarded the maintenance data from its FRONTEND license to the Pool Client - on the server "slot 0" you can see "pool checkouts: 1".
   On the Pool Client, "slot 0" contains the same FRONTEND license as on the server.

3. To demonstrate BACKEND license checkouts, server and client were configured to MCILIB mode (see `PBI=MCILIB` in the example configuration given in **License Pool Setup**).

   The user manually switched to GTL mode with **SYStem.CONFIG DEBUGPORT GTL0** on the TRACE32 command line of both TRACE32 instances.

4. To see different BACKEND licenses, the user then issued the command **SYStem.CPU CORTEXA7** on the Pool Server, and **SYStem.CPU CORTEXM0** on the Pool Client.

5. After this preparation, the user issued **SYStem.Mode Up** on the Pool Server.

   This caused the RLM Floating License Server to check out the BACKEND license 't32.cortexar.gtl' shown in Pool Server "slot 2".

6. Then the user entered **SYStem.Mode Up** on the Pool Client.

   The Pool Client therefore requested the BACKEND license 't32.cortexm.gtl' from the Pool Server.

   The Pool Server obtained the BACKEND license via RLM and forwarded it to the Pool Client (you can see this in "slot 3", notice the "pool checkouts: 1" again).

   The Pool Client shows the BACKEND license in its own "slot 1".

Note

- Floating Licenses checked-out from the Floating License Server (RLM) by the TRACE32-built-in Pool Server are only checked-in when the Pool Server terminates.
  At the moment this is valid for all floating license types.
- The TRACE32 **LICENSE.List** window shows a list of all checked-out licenses at the time the window was opened. This window does not update automatically, so please close and re-open it, if you expect any changes.
- The operating system keeps the TCP port blocked for several seconds (or longer!) if you terminate an application that listens on it. This is to prevent any outstanding in-transit data packets from confusing a newly started application that wants to bind and listen to the same port.

  Unfortunately this mechanism blocks a port, even if an application has properly closed the port.

  To minimize any delay caused by this 'blocking', and to allow the new TRACE32 instance to immediately re-use the same POOLPORT, please close all Pool Clients before you close the Pool Server.

In case of additional questions, please contact our support team at **support@lauterbach.com**.

License Roaming

RLM Roaming offers users equipped with floating licenses the option of temporarily using an off-site license.

With RLM Roaming, users can access licensed software beyond the boundaries of their network environment, enabling them to move between on-site and off-site working scenarios, while respecting license agreements.
Once the license is roamed to the disconnected system, the license will behave as if it were a node-locked.

The TRACE32 license includes a roaming feature that allows users to extract the license and use it temporarily on the same machine during a maximum of 10 days. However, please note that this roaming capability is available and can only be used for the same version of TRACE32 that was originally requested. It is important to ensure compatibility with the version of TRACE32 installed on the original machine to avoid any licensing issues.

There are two approaches to obtaining roaming licenses for TRACE32 users: one is to set the Windows environment variable **RLM_ROAM**, while the other is to use the command **LICENSE.REQuest** command with the **/ROAM** option.
Both configurations require a *<value>* parameter that defines the number of days the roaming license will be available. Setting the parameter *<value>* to 1 indicates the end of the next day, not the end of the current day.

These configurations will be described in more detail in the following.

Using the environment variable

The **LAUTERBACH_ROAM** environment variable serves a similar purpose to **RLM_ROAM**, and it is preferable to use it.

In order to effectively enable license roaming, this section outlines the steps that can be taken:

1. Set the environment variable

   ```
   SET LAUTERBACH_ROAM=<value>
   ```

   or

   ```
   SET RLM_ROAM=<value>
   ```

   where *<value>* is definable in the range 1 to 10

2. Start TRACE32 in the wanted configuration, ensuring that there is an active connection to the RLM server at least once. This is essential so that license can be checked out from the RLM server for the host in the ROAM state.

Once the roaming license has been obtained, it can be used independently without the need to connect to the RLM server.

After the defined amount of days, the license will automatically be rechecked in to the RLM server.

Additionally, it is possible to return the license before the expiration date by setting the environment variable to -1, when the connection to the license server is available.

1. Set the environment variable

```
SET LAUTERBACH_ROAM=-1
```

or

```
SET RLM_ROAM=-1
```

2. Start TRACE32 in the wanted configuration, ensuring that there is an active connection to the RLM server to check in needed licenses to RLM-Server.

Using LICENSE.REQuest

The following TRACE32 command is used to perform temporary “roam” with a floating license within our software infrastructure:

LICENSE.REQuest.plain *<product> <version>* **/ROAM** *<value>* Request roaming license

The roaming duration is controlled by the value specified with the **/ROAM** <*value*> option. The parameter *<value>* is configurable within the defined range of 1 to 10, with 10 serving as the maximum allowable value for roaming licenses.
And note that *<value>* is an integer, in which format it is presented with a dot after the number.

Roaming licenses can only be enabled on the condition that there is an existing connection to the RLM server. Also, licenses can only be checked out from the RLM server as roaming licenses if they have not already been checked out from the RLM server for the host.
When starting TRACE32 for TriCore-GTL, for example, licenses are required for both the GTL back end and the TriCore front end. A problem arises when the front end license is checked out upon starting TRACE32, thus preventing the request of the license product “`t32.frontend.tricore`” for roaming.
Therefore, we recommend using TRACE32 in simulator mode, as it checks the license after the first **Go/Step**.

To illustrate the operation of these considerations, here are the steps to follow:

1. Start TRACE32 in the desired configuration with the regular checkout of RLM licenses and consult **LICENSE.List**. Otherwise, this step can be skipped in case you are aware of the licenses you intend to request for roaming.

2. Start TRACE32 simulator with connection to the RLM server.

3. Request the needed licenses

```
LICENSE.REQuest.plain <product> <version> /ROAM <value>
```

4. Quit the PowerView simulator.

5. Optional: start TRACE32 in the required configuration, so that the **LICENSE.List** is updated to show that all licenses are now marked with “ROAM”.

NOTE:	Roaming licenses are of higher priority than similar licenses from the RLM server. So even if TRACE32 starts with a connection to the RLM server, roaming licenses will be used.

In addition, since TRACE32 nightly build 167005 or release 09.2024 the value specified with the **/ROAM** option can be set to -1., which effectively disables roaming for the license to which it is applied, if this is desired. In this case, the license will be automatically checked in to the RLM server and will not be available for the remaining days previously requested.

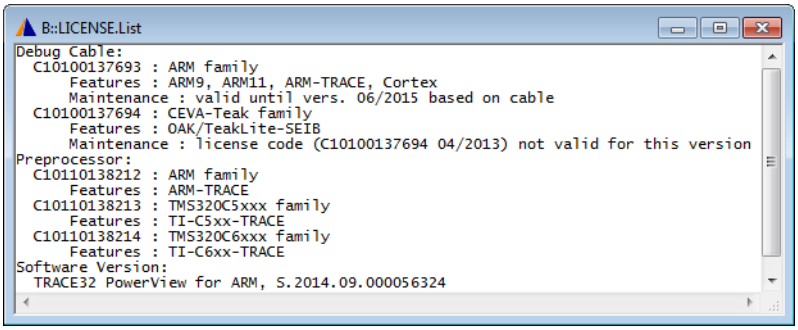
Display a License List

LICENSE.List

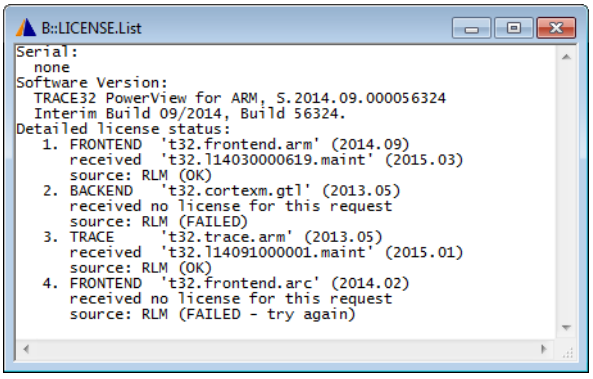
Opens a window which shows all **serial numbers** and corresponding **maintenance contracts** of your debugging product.

If you are using a In-Circuit Debugger, the window shows also the **feature keys** stored in your Debug Cable, Nexus Adapter or Preprocessor.

Example 1: **LICENSE.List** window for a setup with a debug cable and preprocessor:



Example 2: **LICENSE.List** window for a software-only setup using **LICENSE.REQuest**:



Request a License from TRACE32

Format 1:	**LICENSE.REQuest.plain** *<product>* [*<version>*] [**/***<option>*]
Format 2:	**LICENSE.REQuest.***<sub_cmd>* [**/***<option>*]
<sub_cmd>:	**FRONTEND** | **INTEGRATOR** | **MULTICORE** | **SIMULATOR** | **BACKEND** | **TRACE**
<option>:	**ROAM** *<value>* **COUNT** *<value>*

Requests a specific license from TRACE32. If the requested license is not yet available to TRACE32, then the license is checked out from an RLM server. The checked-out license is then blocked for the duration of the TRACE32 session.

You can view the licenses used by TRACE32 in the **LICENSE.List** window. To check the state of the license in a PRACTICE script (*.cmm), use the function **LICENSE.GRANTED()**.

BACKEND	Requests a license for the currently selected backend.
FRONTEND	Requests a frontend license for the current architecture and version.
INTEGRATION	Requests a license for the currently selected third party integration.
MULTICORE	Requests a multicore license for the current software version.
SIMULATOR	Requests a frontend license for the current architecture and version.
TRACE	Requests a trace license for the current architecture and version.
plain	License request for a particular *<product>* and *<version>*.
<product>	License product name as a string, e.g. as given in a lauterbach-*.lic file. For example: “t32.trace.x86”
<version>	License version as a string, e.g. as given in a lauterbach-*.lic file. For example: “2013.05” If the version string is empty, e.g. “”, then TRACE32 will try to auto-fill in the version string, based on the product type.
ROAM *<value>*	Requests the license for roaming. This means, TRACE32 can be used for some days without any connection to the license server. *<value>* defines the number of full days from now. *<value>* can therefore be set to a value between 1. and 10. . To return the roamed license before the allocated time, use the option by passing -1. as the value. This is supported by TRACE32 release 09.2024 or newer. For more information, refer to “Roaming License, floatinglicense.pdf, p xx”.
COUNT *<value>*	Allows the checked-out of multiple licences for a product, to take account of complex multi-core setups.

Please note that it is possible to request and check out Lauterbach licenses (if the license server has them) that are not required to run the current TRACE32 version. This is convenient for testing, e.g. to make sure a particular license is available on the license server.

LICENSE.GRANTED(*<product>*,*<version>*)

Returns an integer value that reflects the current license state of a product-version combination.

Parameter and Description:

<product>	**Parameter Type**: String. License product name, e.g. as given in a lauterbach-*.lic file. For example: “t32.trace.x86”
<version>	**Parameter Type**: String. License version, e.g. as given in a lauterbach-*.lic file. For example: “2013.05” If the version string is empty, e.g. “”, then TRACE32 will try to auto-fill in the version string, based on the product type.

Return Value Type: Decimal value.

Return Value and Description:

0	License found.
1	License not found.
2	License temporarily not available.
3	License permanently not available.
16	*<product>* name was an empty string.

Example:

```
PRINT LICENSE.GRANTED("t32.trace.x86","2013.05")
```

PRACTICE script example

```
// Test for LICENSE commands and functions.

  RESet
  AREA.CLEAR

  GOSUB REQUEST_LICENSE t32.frontend.arm
  GOSUB REQUEST_LICENSE t32.cortexm.gtl
  GOSUB REQUEST_LICENSE t32.trace.arm
  GOSUB REQUEST_LICENSE t32.frontend.arc 2014.02

  WinCLEAR
  WinPOS  0.  0. 60. 21.
  LICENSE.List
  WinPOS  0. 25. 60. 12.
  AREA

ENDDO

REQUEST_LICENSE:
  LOCAL &prodname &prodver &licstate &licinfo
  ENTRY &prodname &prodver
  LICENSE.REQuest "&prodname" "&prodver"
  &licstate=LICENSE.GRANTED("&prodname","&prodver")
  if &licstate==0.
     &licinfo="checkout successful"
  if &licstate==1.
     &licinfo="license not found"
  if &licstate==2.
     &licinfo="no free license, try again"
  if &licstate==3.
     &licinfo="no such license available"
  PRINT "License product='&prodname' version='&prodver'"
  PRINT "         status: &licinfo"
RETURN
```

Acknowledgments

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)