Troubleshooting

This section lists solutions to a set of possible errors which can happen when using the FCI.

Hint

Further help is provided in the troubleshooting page of the manual shipped with your robot.

Cannot boot realtime kernel because of "Invalid Signature"

When you have successfully installed the real-time kernel and try to boot it you might encounter that Linux is not booting at all. This can happen when you have installed Ubuntu alongside with Windows (e.g. Dual Boot). Usually then the UEFI boot loader will have *Secure Boot* activated, which will not allow the unsigned real-time kernel to load.

The easiest solution is to **disable "Secure Boot"** in your boot loader. This highly depends on your system, but usually you can enter the boot loader by pressing F2, F3, F12 or DEL key during boot.

Running a libfranka executable fails with "Connection timeout"

This error occurs if `libfranka` cannot connect to the robot at all. Please check that:

- Using a robot with system version 4.2.0 or higher requires to enable the FCI mode. To do that open Desk -> expand the menu in the sidebar -> press *'Activate FCI'*. Further information about Single Point of Control (SPoC) can be found in the manual shipped with the robot.
- Your workstation is directly connected to Control, not the LAN port of the Arm (see Network).
- The robot can be reached from your workstation (see Robot is not reachable).
- The FCI feature file is installed on the robot (see "Settings -> System -> Installed Features"). If you do not have this feature, please install it. Please reach out to *support@franka.de* with your serial number if you need to obtain the feature.

Motion stopped due to discontinuities or `communication_constraints_violation`

If the difference between commanded values in subsequent time steps is too large, then the motion is stopped with a discontinuity error such as `joint_motion_generator_velocity_discontinuity`. See if the commanded values do not exceed the limits.

Discontinuities can occur if your code commands actual jumps to the robot, but also because of network packet losses. This is also the reason for `communication_constraints_violation` errors. If the issue occurs even when using the provided examples, the problem is most likely related to overall communication quality. To ensure best performance, please check the following:

- All source code is compiled with optimizations (`-DCMAKE_BUILD_TYPE=Release`). If you installed `libfranka` and `franka_ros` from the ROS repositories, this is already the case for these projects. However, your own source code needs to be compiled with optimizations as well.
- Connect your PC directly to Control, without using any intermediate switches. The network setup instructions describe how to do this.
- Verify your network connection by executing the network bandwidth, delay and jitter test.
- `franka::Robot` is instantiated with `RealtimeConfig::kEnforce`. This is the default if no `RealtimeConfig` is explicitly specified in the constructor.
- Power saving features are disabled (cpu frequency scaling, power saving mode, laptop on battery, BIOS power saving features, etc.). See Disabling CPU frequency scaling.

Disabling CPU frequency scaling

CPUs are often configured to use a lower operating frequency when under a light load in order to conserve power. We recommend to disable this feature as it leads to higher latencies when using `libfranka`. To check and modify the power saving mode, install the `cpufrequtils` package:

```
sudo apt install cpufrequtils
```

Run `cpufreq-info` to see available “governors” and the current CPU frequency. Here is an example output:

```
$ cpufreq-info
...
analyzing CPU 0:
  driver: intel_pstate
  CPUs which run at the same hardware frequency: 0
  CPUs which need to have their frequency coordinated by software: 0
  maximum transition latency: 0.97 ms.
  hardware limits: 400 MHz - 3.00 GHz
  available cpufreq governors: performance, powersave
  current policy: frequency should be within 400 MHz and 3.00 GHz.
                  The governor "powersave" may decide which speed to use
                  within this range.
  current CPU frequency is 500 MHz.
...
```

In this example, the maximum frequency is 3 GHz, but the current one is 500 Mhz due to the `powersave` policy. In this case we can benefit by setting the governor to `performance`.

To change this setting using the Ubuntu GUI, install the `indicator-cpufreq` package. A widget in the top bar of the Unity user interface should allow you to set the current policy.

To change this setting using the terminal, execute the following commands:

```
sudo systemctl disable ondemand
sudo systemctl enable cpufrequtils
sudo sh -c 'echo "GOVERNOR=performance" > /etc/default/cpufrequtils'
sudo systemctl daemon-reload && sudo systemctl restart cpufrequtils
```

They will disable the `ondemand` CPU scaling daemon, create a `/etc/default/cpufrequtils` configuration file, and then restart the `cpufrequtils` service.

After enabling the `performance` governor, the `cpufreq-info` results are:

```
$ cpufreq-info
...
analyzing CPU 0:
  driver: intel_pstate
  CPUs which run at the same hardware frequency: 0
  CPUs which need to have their frequency coordinated by software: 0
  maximum transition latency: 0.97 ms.
  hardware limits: 400 MHz - 3.00 GHz
  available cpufreq governors: performance, powersave
  current policy: frequency should be within 400 MHz and 3.00 GHz.
                  The governor "performance" may decide which speed to use
                  within this range.
  current CPU frequency is 2.83 GHz.
...
```

Now the example output shows a CPU frequency close to the maximum one. You can also directly verify the current governor using the `cpufreq-info -p` command.

Robot is not reachable

Try to ping the robot using the following command:

```
ping <fci-ip>
```

If this command fails, the robot is not properly connected to the network, or the IP address is not correctly assigned during the setup phase. Please set up the network according to the documents sent with your robot.

Running a libfranka executable fails with “UDP receive: Timeout”

This error occurs if the robot state can not be received by `libfranka`. Please check that your workstation’s firewall does not block incoming UDP packets (see `sudo iptables -L`).

Network bandwidth, delay and jitter test

In order to evaluate the network performance of your (possible) control PC we’ve developed two tests. The first, a ping test, can be executed without the need of libfranka or franka_ros installed on your system. If your system passes the first ping test, you can run the advanced UDP network performance analysis.

Simple ping tests

The following command will simulate a network load which is equivalent to a scenario where the robot is controlled by the FCI:

```
sudo ping <fci-ip> -i 0.001 -D -c 10000 -s 1200
```

Example output:

```
PING <fci-ip> 1200(1228) bytes of data.
[1500982522.977579] 1208 bytes from <fci-ip>: icmp_seq=1 ttl=64 time=0.279 ms
[1500982522.978423] 1208 bytes from <fci-ip>: icmp_seq=2 ttl=64 time=0.224 ms
[1500982522.979434] 1208 bytes from <fci-ip>: icmp_seq=3 ttl=64 time=0.196 ms
[1500982522.980482] 1208 bytes from <fci-ip>: icmp_seq=4 ttl=64 time=0.243 ms
....
[1500982533.034267] 1208 bytes from <fci-ip>: icmp_seq=9999 ttl=64 time=0.236 ms
[1500982533.035211] 1208 bytes from <fci-ip>: icmp_seq=10000 ttl=64 time=0.203 ms

--- <fci-ip> ping statistics ---
10000 packets transmitted, 10000 received, 0% packet loss, time 10057ms
rtt min/avg/max/mdev = 0.147/0.240/0.502/0.038 ms
```

The example result shows an average round-trip time of 0.24 ms and a maximum round-trip time of 0.5 ms. The standard deviation *mdev* is around 0.04 ms. As explained in the network requirements section it must be guaranteed that the sum of the round-trip time and the execution time of the motion generator or control loop is **less than 1 ms**. If this constraint is violated for a cycle, the received packet is dropped by the FCI.

If the round-trip time is long even with a direct connection, consider purchasing a separate, high performance PCI-Express network card for your workstation PC. See if there are dedicated drivers for your network card, these usually offer better performance. Lastly, the CPU can also be a limiting factor for network performance.

Advanced network performance analysis

The `communication_test` executable, which is part of the `libfranka` examples since version 0.5, executes a control loop and provides feedback about the lost robot states as well as the maximum, minimum and average control command success rate.

If you installed the `ros-noetic-libfranka` package or installed `libfranka` from source, you can execute the test with:

```
source /opt/ros/noetic/setup.sh # only needed when installed with ROS
communication_test <fci-ip>
```

If you do not want to install `libfranka` system-wide you can execute the following command inside your build folder after compiling `libfranka` from source:

```
./examples/communication_test <fci-ip>
```

Running a libfranka executable fails with “Incompatible Library Version”

This happens when your version of libfranka is incompatible with the system version of your robot. The error contains the server version of the robot. You can use that number to choose the correct libfranka version for your robot.

Running a libfranka executable fails with “command rejected due to activated safety function!” or “command preempted due to activated safety function!”

This error occurs when a safety function defined in Watchman (the safety configurator area of the Desk webpage) is active. For example, there could be an active safety function limiting the robot’s speed to 0.2 m/s. As this cannot be guaranteed when using FCI, the robot will not move. However, you can still read the robot state. In order to command movements to the robot again, you either need to disable the safety function or delete the corresponding safety rule in Watchman.