FxResourceMonitor Reference

Record the CPU and memory of a FrameworX server and of every runtime process to a log file, to investigate high CPU, memory growth and crashes.

Reference **Troubleshooting** FxResourceMonitor

Available from FrameworX 10.1.6. This page covers Windows.

Overview

FxResourceMonitor is a command-line tool installed with FrameworX. At a fixed interval it records the CPU and memory of the machine and of each FrameworX runtime process: TServer, TRichClient and every TRunModule. Each module and each device channel is identified separately. It also records when runtime processes start and end, with their exit code, so crashes and restarts appear in the log.

Use it to find which process uses the resources, when, and whether usage keeps growing over hours or days. Growth in memory, handles, threads or GDI objects is the usual sign of a leak.

- It never connects to the runtime. It keeps logging while the runtime is slow, hung or restarting, and it does not affect the runtime.
- It is light. Each sample takes about 10 ms of CPU, and the tool uses a few tens of MB of memory. It logs its own CPU and memory on every sample.
- CPU and memory values use the same scale as Windows Task Manager.

The tool has no user interface. You start it from a command prompt.

Requirements and Location

Build	File	Requires
.NET Framework 4.8	`<install folder>\FxResourceMonitor.exe`	Windows 8.1 or Windows Server 2012 R2 or later, with .NET Framework 4.8
.NET 10	`<install folder>\net10.0\FxResourceMonitor.exe` or `dotnet <install folder>\net10.0\FxResourceMonitor.dll`	The .NET 10 runtime

The default install folder is `C:\Program Files\Tatsoft\FrameworX\fx-10\`. Both builds monitor both kinds of runtime: .NET Framework 4.8 processes (`TServer.exe`, `TRunModule.exe`, `TRichClient.exe`) and Multiplatform processes started as `dotnet TServer.dll` or `dotnet TRunModule.dll`.

Run it from an Administrator command prompt when the runtime runs as a Windows service or under another user account. Without Administrator rights, processes of other accounts are still listed with CPU, memory, threads and handles, but not with their command line, exit code, I/O or memory detail. A .NET 10 process started as `dotnet` by another account cannot be identified at all, and a NOTE line names it. The log header shows whether the tool ran as Administrator.

You can copy `FxResourceMonitor.exe` alone to a machine with an older FrameworX version. It then writes its logs to a `FxResMonLogs` folder next to itself.

Start and Stop

```
FxResourceMonitor.exe [options]
```

All options are optional. Write them as `/key:value`, `-key value` or `--key=value`. Run `FxResourceMonitor.exe /help` for the full list.

To stop the tool, press Ctrl+C, close the window, or let `/duration` end the run. The tool then writes a STOP line and closes the file. Every sample is written to disk immediately, so a forced stop loses at most the sample in progress.

The tool does not restart after a reboot unless you register it with `/install`. See Start at Every Boot below.

Options

Option	Default	Description
`/interval:<seconds>`	10	Time between samples, from 1 to 3600.
`/out:<folder>`	TraceLogs folder	Folder for the log files. See Where the Logs Are Written.

/procs:<name1,name2,...>	TServer, TRichClient, TRunModule	Process names to follow, matched exactly. Add names as needed, for example TReportServer or TStartup. Processes started as dotnet <Name>.dll are recognized automatically.
/duration:<time>	Until stopped	Stop after a time such as 90s, 30m, 8h or 7d.
/maxmb:<MB>	20	Start a new log file when the current one reaches this size, from 1 to 2048.
/maxfiles:<n>	100	Keep at most n log files of this machine in the log folder. The oldest are deleted. 0 keeps all.
/inventory:<minutes>	60	Write the list of monitored processes, with their full command lines, every n minutes. 0 writes it only at the start and at the top of each new file.
/threads:<n>	3	Number of THREAD lines written for a hot process. 0 turns thread detail off.
/hotcpu:<%>	50	CPU, as a percentage of one core, that makes a process hot.
/tcpevery:<n>	1	Count TCP connections every n samples. 0 turns the count off.
/dump	Off	Write memory dumps when a trigger is met. See Memory Dumps.
/analyze[:<folder or file>]	Log folder	Summarize existing logs and exit. See Log Analysis.
/install, /uninstall	None	Register or remove the tool as a task that starts at every boot.
/quiet	Off	No live status line in the console.
/help	None	Show the option list.

With the defaults, the log files use at most about 2 GB of disk (100 files of 20 MB).

Where the Logs Are Written

By default the tool writes to the FrameworX TraceLogs folder, next to the runtime's own trace files: C:\Users\Public\Documents\FrameworX\TraceLogs\. With /out, it writes only to the folder you give.

The tool writes to a FxResMonLogs folder next to itself, and says why on the console, when the FrameworX files it uses to find TraceLogs are not next to it, or when the TraceLogs folder is not writable. As a last resort it uses the system temp folder.

A runtime configured to write its traces to ProgramData writes them to ProgramData\...\Logs. The monitor still writes to TraceLogs unless you give /out.

File name: FxResMon_<MACHINE>_<yyyyMMdd_HHmmss>_<process id>_<part>.log, for example FxResMon_SERVER01_20260924_155903_15896_001.log. A new part (_002, _003) starts when the file reaches /maxmb. Every part repeats the header and the list of processes, so each file can be read on its own. Files are never overwritten.

What the Log Contains

The log is UTF-8 text with one record per line. Fields are separated by ; and use a decimal point. Lines starting with # are the header, which includes a # FIELDS line describing every record type. You can open the file in Excel, process it with a script, or give it to an AI assistant.

The header records the tool version and build, the FrameworX version and log folder, the start time and time zone, the machine, user and Administrator status, the operating system, the .NET version, the CPU model, cores and caches, installed and free memory, boot time and uptime, free space on the log disk, and the settings used.

Record	Written	Content
SYS	Every sample	The machine: CPU busy, idle, kernel and user percentages, memory load, available and total memory, commit and commit limit, system cache, kernel paged and non-paged memory, process, thread and handle counts, average CPU clock and clock limit, context switches per second, and the monitor's own CPU and memory. A clock limit below the maximum shows thermal or power throttling.
PROC	Every sample, one per process	Process id, name, tag, CPU %, memory in KB (Task Manager Memory column), total, user and kernel CPU seconds, threads, handles, GDI and USER objects, working set and peak, private memory and peak, virtual memory, paged and non-paged pool, page faults per second, I/O rates and operations per second, context switches per second, TCP connections and established TCP connections.
THREAD	While a process is hot	The busiest threads of the process: rank, thread id, CPU % of one core, context switches per second, state, wait reason and thread name.

FOUND, START, INV	Process already running, process started, periodic list	Process id, name, tag, parent process, start time, architecture, host (`exe` or `dotnet`), executable path and full command line.
EXIT	Process ended	Exit code in decimal and hex, exit time, lifetime, total CPU seconds, and the last memory, thread and handle values.
DUMP	A dump was written	Trigger, file, size in MB and duration in seconds.
GAP	Samples missed	The monitor itself missed samples, for example while the machine was stalled or asleep.
NOTE	As needed	Something the monitor could not do, such as access denied to a process.
STOP	End of the run	Reason, number of samples and duration.

The tag identifies a process over time, even when it restarts with another process id. For TRunModule it is the module and device channel, for example `Alarm`, `Dataset` or `Device/Modbus1`. For TServer it is `<solution>:<port>`, for example `MySolution:3101`.

An empty field means the value is not readable with the current permissions, or is a rate on the first sample of a process.

Reading the Values

- **CPU %** is the share of the whole machine over the interval, the scale Task Manager uses. Task Manager truncates it to an integer, so 2.76 shows as 2. One fully busy core equals 100 divided by the number of logical processors, for example 10 % on a 10-processor machine.
- **Memory.** The KB value equals the Task Manager Memory column. Private MB rising steadily over hours is the classic memory-leak signal. . NET processes go up and down as garbage collection runs, so watch whether the lowest values keep rising. Virtual MB matters only for 32-bit (x86) processes.
- **Handles, threads, GDI and USER objects** that keep growing are also leak signals. GDI and USER objects apply to processes with a user interface, such as TRichClient.
- **Context switches per second** show how often a process's threads are scheduled. A mostly idle process with thousands per second and a high kernel CPU share wakes up too often, from short sleeps, timers or polling.
- **Wait reason** on THREAD lines: `DelayExecution` is `Thread.Sleep`, `UserRequest` is a wait on an event or lock, `WrQueue` is the thread pool or I/O completion. Use the thread id to find the thread in a dump or debugger.
- **TCP connections** that keep growing indicate a socket leak.
- **Exit codes:** `0xC0000005` access violation, `0xE0434352` unhandled .NET exception, `0xC00000FD` stack overflow, `0xC0000409` fail-fast or buffer overrun, `0x80131506` .NET runtime fatal error. `1` or `-1` usually means the process ended normally or was killed.

A short run shows the load. Detecting a leak needs hours or days of normal use.

Memory Dumps

Dumps are off by default. With `/dump`, a process that meets a trigger for several consecutive samples gets one dump attempt. The dump includes the heap, which is enough for .NET memory and thread analysis.

Option	Default	Description
`/dumpcpu:<%>`	90	CPU at or above this percentage of one core. 0 disables the CPU trigger.
`/dumpmem:<MB>`	0 (off)	Private memory at or above this many MB.
`/dumpsamples:<n>`	6	Consecutive samples that must meet the trigger (1 minute at a 10-second interval).
`/dumpmax:<n>`	3	Successful dumps per run.
`/dumpdir:<folder>`	Log folder	Folder for the dump files.

- The first 5 minutes of a process do not count toward a trigger, because start-up is busy.
- The process is paused while the dump is written, from under a second to several seconds. A dump takes about the process's private memory on disk. The tool skips the dump, with a NOTE, when less than 1.5 times the private memory plus 512 MB is free.
- A NOTE is written before the dump and a DUMP record after it. With `/dump`, a NOTE at start says whether the dump folder is ready.
- Dumps contain process memory, which can include customer data, connection strings and credentials. Each dump file is readable only by SYSTEM, Administrators and the account running the monitor, even when the log folder is open to everyone. The tool refuses to write a dump through a junction or link in the folder path.
- Run the tool as Administrator to dump a runtime that runs as a service. Log retention (`/maxfiles`) never counts or deletes dumps.

Log Analysis

```
FxResourceMonitor.exe /analyze
FxResourceMonitor.exe /analyze:D:\FxLogs
```

`/analyze` reads the logs of the default log folder, or of the folder or file you give, prints a report and saves it as `FxResMon_Analysis_<stamp>.txt` next to the logs. Logs from several machines in one folder are reported per machine.

The report has one block per process instance, identified by process id and start time, so two clients of the same solution are never merged. Each block shows CPU % average, 95th percentile and maximum, also as a percentage of one core, context switches, and for private memory, handles, threads, GDI and USER objects the lowest value of each 10-minute window with its trend per hour. The first 30 minutes of each process are left out of the trends, because start-up growth is not a leak.

Finding	Condition
MEMORY, HANDLES, THREADS, GDI or USER GROWING	The lowest values keep rising. Needs at least 2 hours of data after start-up.
HIGH CPU	Average at or above 50 % of one core.
CPU PEAKS	95th percentile at or above 90 % of one core.
CRASHED	Exit code starting with 0xC, 0xE or 0x8013, except `0xC000013A` (Ctrl+C or window closed).

The report also lists the dumps written. Short process instances without a finding, such as a module that restarts, are summarized in one line per program.

Start at Every Boot

```
FxResourceMonitor.exe /install [options]
FxResourceMonitor.exe /uninstall
```

Run these from an Administrator command prompt. `/install` creates the scheduled task `FxResourceMonitor`, which runs as SYSTEM at system startup with the other options you give, and starts it immediately. Running `/install` again replaces the options and restarts the monitor. `/uninstall` removes the task. Stopping the task ends the monitor without a STOP line.

Run `/install` from the FrameworX installation folder. Because the task runs the program as SYSTEM, `/install` refuses with exit code 7 when the program, its FrameworX DLLs, the .NET runtime it uses, or any folder above them can be changed by an account other than SYSTEM, Administrators or TrustedInstaller.

Disk Space

Each sample writes about 150 bytes for the machine and about 190 bytes per monitored process. At the default 10-second interval:

Monitored processes	Per day	30 days	Options for 30 days
About 10	About 18 MB	About 0.5 GB	Defaults
102 (TServer, TRichClient and 100 TRunModule)	About 170 MB	About 5 GB	`/duration:30d /maxmb:100 /maxfiles:60`

Halving the interval doubles these numbers. THREAD lines add volume only while a process is hot. If the disk fills up, the tool drops samples, retries at most once a minute and continues in a new file when space is available, without deleting older logs.

Examples

Task	Command
Monitor for one week with the defaults, as Administrator	`FxResourceMonitor.exe /duration:7d`
Include the report server and write to another disk	`FxResourceMonitor.exe /procs:TServer,TRichClient,TRunModule, TReportServer /out:D:\FxLogs`
Sample faster for a short investigation	`FxResourceMonitor.exe /interval:2 /duration:1h`
Dump a runtime module whose private memory stays above 2 GB	`FxResourceMonitor.exe /dump /dumpcpu:0 /dumpmem:2048 /procs: TRunModule`
Start at every boot with a 30-day retention	`FxResourceMonitor.exe /install /maxmb:100 /maxfiles:60`
Summarize the logs	`FxResourceMonitor.exe /analyze`

Exit Codes

Code	Meaning
0	Completed
2	Invalid option, for example a `/dumpdir` that is not a valid folder path
3	Log folder not writable
4	`/install` or `/uninstall` run without Administrator rights
5	Install failed
6	`/analyze` failed
7	Install refused because the program folder can be changed by other accounts

Limitations

- A process that starts and ends between two samples does not appear in the log.
- The tool shows which process uses resources and when, not which part of the code. Once you find a suspect process, take a memory dump or a trace of that process.
- The log contains the machine name, user name and full command lines, including solution file paths and ports. Review it before you send it outside your organization.

In this section...
