Open iT

SIMULATION & ANALYSIS · LICENCE HARVESTING

The Solver-versus-Session Harvesting Guide

Which CAE tools can hand a licence back, and how thresholds are set so a running solve is never touched.

01 What separates a solve from a parked session, and what the licence server can and cannot see

02 The licence managers in scope for automated harvesting, and what that means tool by tool

03 Every threshold in the configuration, what it governs, and the one setting that means *never touch this*

04 Saving work before the licence moves, and the interactive-job case on IBM LSF

05 A rollout order your engineering leads can agree to before anything is enabled

Open iT · License*Analyzer*™ 10.2 Every parameter in this guide is quoted from the Open iT product documentation. Sources on the last page.

01 THE DISTINCTION

A checkout is not a measurement of work

Your licence manager records one event: the seat was taken. A six-hour solve and a results window left open over a long weekend produce the same record. Harvesting is only safe once something else is measuring what happened behind that checkout.

LEVEL 1

Runtime Usage

What the licence manager reports. For Ansys, that is FlexNet polled with `lmutil lmstat`, plus checkout and check-in events from the vendor daemon debug log. It tells you a seat is held. It does not tell you why.

LEVEL 2

True Active Usage

Activity measured at the workstation: CPU utilisation, I/O operations, keyboard strokes and mouse interactions, on Windows, Unix and macOS. This is the layer that can tell a running solve from an idle window.

LEVEL 3

Managed Usage

Acts on what Level 2 measured. An idle application can be suspended — freeing the licence temporarily while preserving the application's state, so the engineer can resume work rather than start over.

> **Why this is the whole safety argument**
>
> A solve holds the CPU and moves data to disk. A parked window does neither, whether or not anyone is logged in. Because the same four signals — CPU, I/O, keyboard, mouse — are what the disabler evaluates, the running solve is the case the mechanism is built to leave alone. Everything in section 03 exists to let you set where that line sits.

The collectors that produce the signal

COLLECTOR	WHAT IT CONTRIBUTES
AM	Keyboard and mouse events, and which application window has focus
FreezeMonitor	Groups process data by user configuration and records status against the activity thresholds — the component that later decides an application has gone idle
AppUsage	Process information
Disabler Pop-up	Collects the engineer's response when an application is found inactive

On Unix and macOS, ProcessAccounting and PerformanceMonitor take the place of some of the Windows components.

02 SCOPE

Which tools can actually hand the licence back

Suspend and resume is not a per-vendor feature. It is a method set against an application entry in the configuration — so what decides whether a licence genuinely returns to the pool is the licence manager the application is served by.

Licence managers named in Level 3's published harvesting integration

LICENCE MANAGER	TYPICAL CAE ESTATE SERVED	HARVESTING
FlexLM / FlexNet Publisher	Ansys (vendor daemon `ansyslmd`) and the large majority of commercial solvers and pre/post tools	**In scope**
LM-X	X-Formation-licensed engineering applications	**In scope**
RLM (Reprise)	Reprise-licensed solvers and meshing tools	**In scope**
IBM LUM	Legacy IBM-licensed engineering applications	**In scope**

Managers documented for usage collection, but not named in that integration

Open iT documents a Level 1 collection configuration for each of these as a licence manager in its own right. Usage measurement is not in question; automated release-back is what needs confirming for your estate before anyone promises it.

LICENCE MANAGER	WHERE IT SHOWS UP IN A CAE ESTATE	HARVESTING
Abaqus / Elan	Abaqus estates served by the Elan licence manager	Confirm
Dassault Systèmes (DSLS)	SIMULIA and CATIA estates on the DS licence server	Confirm
LS-DYNA Licence Manager	LS-DYNA served by its own manager rather than FlexNet	Confirm
Altair Licence Manager	HyperWorks, OptiStruct and the Altair units model	Confirm
MATLAB Licence Manager	MATLAB and Simulink alongside the solver estate	Confirm
Beta LM	BETA CAE pre/post processing	Confirm
Moldex3D	Injection-moulding simulation	Confirm

Many vendors ship more than one licensing option. Check which manager actually serves your seats — the vendor name does not decide this, the manager does.

The four actions available against an idle application

`log`	Record the inactivity and do nothing else. This is the observation mode you run first.
`standby`	Prompt the engineer, who can choose to carry on using the application.
`suspend`	Release the licence while preserving the application's state, so work resumes rather than restarts.
`terminate`	Close the application, optionally offering a restart.

03 THRESHOLDS

Setting the line so a solve is never touched

Harvesting is configured per application in `cold_master.conf`. These parameters decide what counts as work, how long idleness must last, and who is exempt.

Timing — how long, and how much warning

PARAMETER	WHAT IT GOVERNS
`timeout`	How long the process must be inactive before it is considered for disabling. ISO-8601, e.g. `P45M`.
`irq`	The window in which the engineer is warned through the disabler interface first, e.g. `P15M`.
`suspend-timeout`	How long a suspended application stays suspended. Default `P6H`.
`suspend-timeout-action`	What happens when that expires: `resume`, `terminate` or `none`.

Activity — what counts as work

Four signals are evaluated, each with a threshold and a weight. Because CPU and I/O are among them, machine work registers as activity even when nobody is at the keyboard — the unattended-solve case exactly.

SIGNAL	THRESHOLD	DOCUMENTED DEFAULT
`keyboard`	Keyboard activity, in events per second, with a weight	Site-specific
`mouse`	Minimum mouse events per second, with a weight	Site-specific
`cpu`	CPU load threshold, with a weight	Threshold `1.5`, weight `1`
`i/o`	I/O activity, in bytes transferred per second, with a weight	Threshold `1.5`, weight `1`
`disable-threshold`	The global value the CPU and I/O weights are summed against	`4`
`cpu-formula-v2`	Uses percentage CPU utilisation in the calculation	`yes`

The setting that means never touch this

The priorities section changes the timeout per user or group, and one of its values removes the application from harvesting altogether:

`total`	**Never disabled.** Inactivity is logged and nothing else happens. This is where validation runs, certification work and anyone you have agreed to exempt belong.
`high`	Timeout doubled.
`normal`	The configured timeout.
`low`	Timeout halved.

When to harvest, and what to save first

Two things decide whether engineering accepts harvesting at all: that it only fires when the pool is genuinely under pressure, and that the model is safely on disk before the licence moves.

Rules — harvest only when it is warranted

Conditions are evaluated with the usual operators against four keys, so harvesting can be conditional rather than constant:

KEY	USE IT FOR
`license-use`	Percentage of licences in use — so nothing is reclaimed while the pool has spare seats
`count`	Number of processes
`wday`	Day of the week, 1–7
`time`	Time of day with timezone — also how a schedule is set for vendors billing by calendar day

Custom actions — saving before release

Suspend preserves the application's state. For teams that want the model written to disk as well, a custom action runs before the licence is released.

PARAMETER	WHAT IT GOVERNS
`command`	The binary or executable to run
`when-to-run`	`pre` — before the optimisation acts — or `post`, after it
`condition`	`once`, `all`, `parent`, `child`, or `exe=<pattern>` to target a specific executable
`timeout`	How long the action may take. Default `P5M`.

A save step is a `pre` action scoped with `exe=` to the application you are protecting. Give it a timeout that a large model can actually finish inside — the default five minutes is a starting point, not a specification for a multi-gigabyte assembly.

> **Built-in save script published in 10.2**
>
> **Hexagon Smart 3D.** The script activates the target window, drives the exit dialog, generates a timestamped filename, writes the project data to a designated directory, and then lets the application complete its normal shutdown. That sequence is the pattern to copy for any other tool: bring the window forward, drive its own save path, write somewhere predictable, and only then let the licence go.

Interactive jobs on IBM LSF

An interactive job holds its licence for the length of the job slot rather than the length of the work, which is why the block never shows up in a peak-concurrency figure. Open iT handles this case by having FreezeMonitor identify which executables were launched inside the job slot, using the job starter script `openit_job_starter.sh` from `/opt/openit/bin`.

> **One requirement worth knowing before you plan for it**
>
> The documented LSF procedure is **Unix only**. Plan the interactive-job work against your Unix estate, and confirm anything else with your implementation engineer rather than assuming parity.

05 ROLLOUT

The order that keeps engineering on side

Every objection to harvesting is the same objection: something of mine will be closed while I am using it. The sequence below answers it with evidence before anything is ever suspended.

1. **Run in `log` for a full working cycle.**
   Nothing is suspended, nothing is terminated. You get a record of which sessions would have been flagged, which is the only credible basis for the conversation that follows.

2. **Review the log with the engineering leads, not the licence owner.**
   The people who know which of those sessions were real work are the ones who run the models. Their edits to the list become your exemptions.

3. **Set `priority: total` on protected work.**
   Validation runs, certification work, anything a programme milestone depends on. Documented, agreed, and never disabled.

4. **Agree `timeout` and `irq` in writing.**
   How long idle is idle, and how much warning an engineer gets. These two numbers are the whole negotiation — settle them once and stop relitigating.

5. **Enable `standby` before `suspend`.**
   The prompt appears and the engineer can decline. Nobody loses anything, and you find out how often the detection is right.

6. **Move to `suspend`, with a save action where it matters.**
   Set `suspend-timeout` and its action deliberately: `resume` behaves very differently from `terminate` at the end of six hours.

7. **Gate it with `license-use`.**
   Harvest when the pool is under pressure, not when there are seats spare. This single rule removes most of the remaining political objection.

8. **Watch the suspend-action counts.**
   An unexpected jump means a threshold is mis-set for that application, not that engineers changed their habits.

Agree these five before go-live

- [ ] The exemption list, and who may add to it
- [ ] The `timeout` and `irq` values, per application family rather than globally
- [ ] Whether a save action runs, and where it writes
- [ ] What happens at the end of `suspend-timeout`
- [ ] Who reviews the first month of harvesting data, and when

06 BASIS

What this guide claims, and what it does not

> **Every parameter, default and capability above is quoted from Open iT's product documentation for version 10.2.** Where the documentation does not publish a value, this guide says site-specific rather than supplying a number.

Three things are deliberately absent:

- ☐ **No recovery percentage.** What a given estate gets back depends on its own usage shape. Measure it before anyone budgets against it.
- ☐ **No threshold recommendations.** A meshing session and an overnight solve idle differently. The values belong to your applications and your engineering leads, which is why section 05 starts in observation mode.
- ☐ **No cost figures.** Seats recovered are a licensing outcome, not a saving, until a renewal decision is actually taken against them.

Sources

- License*Analyzer*™ Level 2 — activity signals, collectors, supported operating systems. docs.openit.com/10.2/license_analyzer/level2/
- License*Analyzer*™ Level 3 — suspend and state preservation, supported licence managers. docs.openit.com/10.2/license_analyzer/level3/
- Optimizing Application Usage — `cold_master.conf` parameters, thresholds, priorities, rules, custom actions. docs.openit.com/10.2/license_analyzer/level3/optimizing_application_usage/
- Disabler Process — suspend, terminate and standby prompts. docs.openit.com/10.2/license_analyzer/level3/disabler_process/
- Built-in Save Scripts — Hexagon Smart 3D. docs.openit.com/10.2/license_analyzer/level3/save_scripts/hexagon_save_script/
- Optimizing Interactive Jobs in LSF — job starter script, Unix requirement. docs.openit.com/10.2/license_analyzer/level3/optimizing_interactive_lsf_jobs/
- Licence Harvesting Schedule for Daily Entitlements — scheduling through the rules and priorities sections. docs.openit.com/10.2/license_analyzer/level3/license_harvesting_schedule_for_daily_entitlements/
- FlexNet Licence Manager — `lmutil lmstat` polling, vendor daemon log collection, options-file parsing. docs.openit.com/10.2/license_analyzer/level1/flexnet/

NEXT STEP

A utilisation snapshot measures your own estate against every distinction in this guide — before any threshold is set.

Talk to a licensing specialist · openit.com

Open iT