QUANTO USER GUIDE

Installation and folders

Program Files, add-ins, Local AppData, Public Documents, .qtb / .qtm / layouts, licenses.

This guide is for IT and lab administrators. It lists every folder Quanto uses after setup, what may be shared on a network, and what can be deleted safely.

Quanto is a **64-bit Windows desktop** application (WPF, .NET 10). It is not a Windows service and does not need IIS, SQL Server, or a dedicated application server. Analysts run it on a workstation. A file server is optional for raw data, methods, licenses, and finished batches.

What the installer does

The setup program (`Quanto-<version>-setup.exe`) is an Inno Setup package.

Item	Value
Install location	`C:\Program Files\Quanto` (the wizard does not offer another folder)
Architecture	64-bit Windows only
Start Menu	Quanto group with the app, website, and uninstall
Desktop	Quanto shortcut
Folders created	`C:\Users\Public\Documents\Quanto` and `%AppData%\Quanto` (Roaming)
File associations	None. Drop a `.qtb` , `.qtm` , `.qtl` , or `.lic` onto the running window or the exe

Uninstall removes `C:\Program Files\Quanto` . It does **not** remove user settings, layouts, licenses, methods, or batch files.

The installed `Quanto.exe` is a small launcher. It starts `Quanto.WPF.exe` . The WPF host is a framework-dependent .NET 10 Windows app, so PCs need the **.NET 10 Desktop Runtime (x64)** unless your package already includes it.

Agilent MassHunter data that is not already indexed also needs Agilent's converter (see Vendor extras).

Inno Setup parameters

The package is built from `Source\Quanto.WPF.iss` (Inno Setup 6). These choices are fixed in the script — the wizard does not ask for them.

What the script locks in

Setup directive	Value	Effect for IT
`DefaultDirName`	`{pf}\Quanto`	Always installs to `C:\Program Files\Quanto` on 64-bit Windows
`DisableDirPage`	`yes`	No “Choose install folder” page
`DisableProgramGroupPage`	`yes`	No Start Menu folder page; group is always **Quanto**
`ArchitecturesInstallIn64BitMode`	`x64`	64-bit only; refuses 32-bit Windows
`ShowLanguageDialog`	`no`	English only
`Compression`	`lzma2` solid	Smaller setup; normal for desktop apps
Desktop icon	always created	No optional task — every install gets a desktop shortcut
`[Dirs]`	`{commondocs}\Quanto`, `{userappdata}\Quanto`	Creates Public Documents and Roaming AppData folders
Post-install launch	`skipifsilent`	Interactive installs offer “Launch Quanto”; silent installs do **not** start the app
Prerelease banner	build date + 60 days	Welcome / Ready / Finished pages show the expiry text

Uninstall uses the same `AppId`. Upgrading over an existing install replaces `C:\Program Files\Quanto` and keeps user data elsewhere.

Command-line parameters (deploy / SCCM / Intune)

Run elevated. Example setup name: `Quanto-1_8-setup.exe`.

Parameter	Purpose
`/SILENT`	No prompts; shows a progress window
`/VERYSILENT`	No prompts and no progress window (best for software center)
`/SUPPRESSMSGBOXES`	Auto-answer message boxes (use with silent)
`/NORESTART`	Do not reboot even if Windows asks
`/DIR="C:\Program Files\Quanto"`	Override install folder (still allowed even though the dir page is hidden)
`/GROUP="Quanto"`	Override Start Menu group name
`/LOG="C:\Temp\quanto-setup.log"`	Write a detailed Inno log
`/CLOSEAPPLICATIONS`	Close apps that lock install files (prompts unless silent)
`/FORCECLOSEAPPLICATIONS`	Force-close without asking
`/NOCLOSEAPPLICATIONS`	Never auto-close; fail if files are locked

There are **no** optional `[Tasks]` in this script, so `/TASKS=` / `/MERGETASKS=` have nothing to select. There is no `/COMPONENTS=` either — one full install only.

Typical silent upgrade

`bat Quanto-1_8-setup.exe /VERYSILENT /SUPPRESSMSGBOXES /NORESTART /LOG="%TEMP%\quanto-setup.log"`

After a silent install, start Quanto yourself (or via a shortcut). The setup will not launch it when `/SILENT` or `/VERYSILENT` was used.

Folder map

``` C:\Program Files\Quanto\ Program (replace on upgrade) Quanto.exe Launcher Quanto.WPF.exe Main application AddIns\ Vendor and integrator plugins DataProvider\ Read .D / WIFF raw files Integrator\ Peak detection (Quanto, Slice) MethodImporter\ Vendor method import VendorSupportFiles\Agilent\ Agilent native support DLLs VendorSupportFiles\Sciex\ SCIEX native support DLLs Layouts\Default.qtl Shipped workspace preset

C:\Users\Public\Documents\Quanto\ Machine-shared (all users) Config\ Themes.xlsx, Languages.xlsx, Layouts.xlsx License\ Default *.lic folder

%LOCALAPPDATA%\Quanto\ Per-user (this is the live profile) Settings.json Colors, formats, dialogs, MRU, license path appearance-settings.json Theme, language, zoom, density FormSettings_.json Window sizes for this screen layout GridSettings.xml Navigator / filter leftovers Layouts\ Workspace and method-editor layouts BatchReports\ Temporary Excel cache from .qtb / .qtm Recovery\ Crash packages pending-open.txt Second-instance file hand-off
```

\ Chosen by the analyst sample.d / sample.wiff Vendor raw data (do not move after extract) QuantoResults\.qtb Batch (results + chromatograms + method copy) QuantoReports... Exported Excel / PDF

\method.qtm Method package (user-chosen path) ```

`%ProgramData%\Quanto` is **not used**. Do not create it. `%AppData%\Quanto` (Roaming) may exist because the installer creates it; the running app reads and writes **Local** AppData.

Program folder

`C:\Program Files\Quanto`

Replace this folder on every upgrade. Users do not store studies here.

Add-ins

Add-ins are loaded at startup from three subfolders. Extra DLLs next to those folders are also searched (including `VendorSupportFiles`).

Folder	Role	Typical DLLs
`AddIns\DataProvider`	Open vendor data	`AgilentMHData.dll`, `AgilentCSData.dll`, `SciexWiff.dll`
`AddIns\Integrator`	Peak detect / integrate	`Quanto.dll` (default), `Slice.dll`
`AddIns\MethodImporter`	Import vendor methods	`AgilentMHMethod.dll`, `AgilentCSMethod.dll`, `AgilentMassHunterDataMethod.dll`, `SciexWiffMethod.dll`
`AddIns\VendorSupportFiles\Agilent`	Agilent native libraries	`AgilentVendor.dll` and related
`AddIns\VendorSupportFiles\Sciex`	SCIEX native libraries	Clearcore / WIFF support

Do not mix add-in versions from different Quanto builds. After an upgrade, keep the whole `AddIns` tree from that build.

Shipped layouts

`C:\Program Files\Quanto\Layouts\Default.qtl` is the factory workspace. User copies and custom presets live under Local AppData (below).

Machine-shared folders (all users on the PC)

`C:\Users\Public\Documents\Quanto`

Subfolder	Contents	Notes
`Config\`	`Themes.xlsx`, `Languages.xlsx`, `Layouts.xlsx`	Fast appearance catalog. Seeded on first run if empty. Edit here to add a language or theme for every user on the PC.
`License\`	`*.lic`, optional `withoutlicense.txt`	Default license folder. 21-day install grace; data older than 30 days is free; otherwise the instrument serial must appear in a `.lic` file.

A site can point Settings → license folder at a **network share** so every PC reads the same `*.lic` files. The path is stored in that user's `Settings.json` (`LicenseFolder`).

Per-user profile (settings and layouts)

`%LOCALAPPDATA%\Quanto`
Typical path: `C:\Users\<user>\AppData\Local\Quanto`

This is the folder to back up, roam (if you must), or delete to reset one user. Hold **Shift** while the splash is visible to recycle settings, layouts, and/or licenses without hunting these paths.

File or folder	What it is
`Settings.json`	Formats, colors, plot titles, dialog sizes, quick-access folders, last batch/method paths, MRU lists, license folder, first-use date
`appearance-settings.json`	Theme (Light/Dark), language, UI zoom, compact/standard/spacious
`FormSettings_<WxH>.json`	Window geometry for the screen size at first launch (example: `FormSettings_3840x2160.json`)
`GridSettings.xml`	Older navigator/filter bits (mostly unused by Fast layouts)
`Layouts\MainWindow\_active\`	Live docking + Fast grid column layouts
`Layouts\MainWindow\Presets\*.qtl`	Saved workspace presets (zip)
`Layouts\MainWindow\Autos\`	Autosaved workspaces
`Layouts\MethodEditor\`	Method editor docking presets (`*.qtl`) and `_active`
`BatchReports\<hash>\`	Unpacked report workbooks for the open batch. Safe to delete when Quanto is closed
`Recovery\`	Last crash packages (`Quanto-Recovery-<pc>-<stamp>.zip`), max five
`pending-open.txt`	Brief hand-off when a second Quanto is started with a `.qtb`

Fast grid files under `_active\Grid\` look like `Results-FastLayout-ResultsGridView.json`. Docking files are `DockingLayout_*.json`.

Roaming profiles: keep this tree small (settings + layouts). Do not roam `BatchReports` or `Recovery`. Chromatograms belong in the study folder, not in the profile.

Study / batch folder (the important data)

Quanto does not pick a company-wide data root. The analyst chooses a **data path** (local disk, instrument PC, or share). The batch file is stored next to the raw files:

`D:\Studies\Pesticide-2026-03-15\ Cal-01.d Cal-02.d Sample-01.d ... QuantoResults\ 2026-03-15.qtb QuantoReports\ 2026-03-15_Batch_2026-03-15_18-40-00\`

Item	Role
Vendor files (`.d` folders, `.wiff`)	Raw acquisition. Extract reads them. Keep them; a `.qtb` does **not** replace the raw files if you need to re-extract.
`QuantoResults\<name>.qtb`	Zip package: `BatchInfo.json` , method tables (`method/*.tsv`), result tables (`result/*.tsv`), chromatograms (`Time.chr` , `Original.chr` , `Modified.chr` , TIC `*.chr`), `Integrations.int` , optional `reports/*.xlsx` .
`QuantoReports\`	Excel / PDF written by Report export. Default root is the same data path.

Standard location for a batch is **always** `<data path>\QuantoResults\<name>.qtb` . A copy on the desktop still remembers the original data path inside `BatchInfo.json` , so open it only when that data folder is still reachable.

Agilent MassHunter extract also needs an index inside each sample:

`sample.d\AcqData\MSTree2.bin`

If that file is missing, Quanto looks for Agilent `TDAConverterConsole.exe` and writes the index **into the sample folder**. The PC account needs write permission on the `.d` folder.

Method files

Kind	Typical path	What is inside
Quanto method	Anywhere the user saves, `yyyy-MM-dd.qtm`	Zip: `MethodInfo.json` , `method/*.tsv` (analytes, quants, doses, types, AcqKeys, stored calibration), optional `workbooks/*.xlsx`
Vendor method	MassHunter / ChemStation / SCIEX folders	Imported, then saved as `.qtm` for daily use

Methods are not stored under Program Files. Put a lab “current methods” share on a **local or fast** disk if many PCs load the same `.qtm` . Loading a `.qtm` is cheap compared with extract.

A saved `.qtb` already contains a copy of the method tables used for that batch.

Layout files

Kind	Extension	Location
Workspace preset	`.qtl`	User `Layouts\MainWindow\Presets`, plus shipped `Layouts\Default.qtl`
Method editor preset	`.qtl`	User `Layouts\MethodEditor\Presets`
Live docking / grids	`.json`	User `Layouts\...\ _active`

A `.qtl` is a zip of the active docking layout plus Fast grid JSON. Users can drop a `.qtl` onto Quanto. Layouts do not contain results.

Permissions

Path	Users need
`C:\Program Files\Quanto`	Read / execute
`C:\Users\Public\Documents\Quanto\License`	Read (write if they install `.lic` files themselves)
`C:\Users\Public\Documents\Quanto\Config`	Read (write only if you edit themes/languages centrally)
`%LOCALAPPDATA%\Quanto`	Full control (created automatically)
Study folder + `QuantoResults`	Read/write
Agilent `.d` folders	Read; **write** if indexed conversion must run on this PC
Shared license folder	Read for analysts; write for the person who drops new `.lic` files

Quanto does not require local Administrator after install.

Antivirus, backup, and roaming

Real-time scanning of vendor `.d` folders and `QuantoResults` during Analyze is a common cause of slow extract and failed conversion. Exclude:

- `C:\Program Files\Quanto`
- Study disks or the instrument data share (or at least `*.d`, `MSTree2.bin`, `*.wiff`, `*.qtb`)
- `%LOCALAPPDATA%\Quanto\BatchReports` if report export is scanned heavily

Backup (in priority order)

1. Study folders (raw data + `QuantoResults\*.qtb` + `QuantoReports`)
2. Method share (`*.qtm`)
3. License folder
4. Optional: `%LOCALAPPDATA%\Quanto` (settings and layouts only)

Safe to delete when Quanto is closed: `BatchReports`, `Recovery` (after you copied a crash zip), `%TEMP%\Quanto`.

Do not delete to "fix" a slow PC: `QuantoResults`, raw `.d` / `.wiff`, or `*.qtm` methods.

Vendor extras

Need	Software / path
Agilent MassHunter extract when `MSTree2.bin` is missing	`C:\Program Files\Agilent\MassHunter\Workstation\Quant\bin\TDAConverterConsole.exe` (MassHunter Quant). Better: enable indexed-data export on the instrument so conversion already happened.
Agilent ChemStation `.D`	Bundled `AgilentCSData` add-in
SCIEX WIFF	Bundled `SciexWiff` add-in + `VendorSupportFiles\Sciex`

Quanto does not install Agilent MassHunter or SCIEX Analyst.

Reset and support

- **Shift** during splash: recycle settings, layouts, and/or licenses (Recycle Bin).
- Crash zip: `%LOCALAPPDATA%\Quanto\Recovery\Quanto-Recovery-<pc>-<stamp>.zip` (settings, layout, batch snippet, logbook).
- In-app logbook is memory-only (about 2500 lines) unless a crash package is written.

Related

- Hardware and performance (IT) — CPU, RAM, SSD, network, batch size, purchase tiers
- Analyst how-to lives in the user guides

QUANTO USER GUIDE

Silent deploy

/VERYSILENT checklist, .NET 10 Desktop x64, no post-install launch, TDA optional.

Checklist for SCCM / Intune / unattended install from the Inno Setup section of Installation and folders.

Prerequisites

Item	Notes
OS	Windows 10/11 **64-bit**
Runtime	**.NET 10 Desktop Runtime (x64)** unless your package already bundles it
Elevation	Run the setup elevated
TDA converter	**Optional** — only for MassHunter `.D` that lack `MSTree2.bin` (Agilent MassHunter Quant). ChemStation and SCIEX do not need it

Typical silent command

Example setup name (match your build artifact):

```
bat Quanto-1_8-setup.exe /VERYSILENT /SUPPRESSMSGBOXES /NORESTART /LOG="%TEMP%\quanto-setup.log"
```

Parameter	Purpose
`/VERYSILENT`	No prompts, no progress window
`/SUPPRESSMSGBOXES`	Auto-answer message boxes
`/NORESTART`	Do not reboot even if Windows asks
`/LOG=...`	Detailed Inno log

Also available: `/SILENT` (progress window), `/DIR=`, `/GROUP=`, `/CLOSEAPPLICATIONS`, `/FORCECLOSEAPPLICATIONS`, `/NOCLOSEAPPLICATIONS`. There are no optional Tasks/Components in this script.

After silent install

- The setup does **not** launch Quanto when `/SILENT` or `/VERYSILENT` was used (`skipifsilent`). Start the app yourself or via a shortcut / software-center step.
- Install location is `C:\Program Files\Quanto` unless `/DIR=` overrides it.
- User profile folders are created on first run under `%LOCALAPPDATA%\Quanto`.

Permissions pointer

Use the permissions table in Installation and folders: Program Files read/execute; Public Documents license/config as needed; Local AppData full control; study folders and Agilent `.D` write when MH indexing must run on the PC.

Related

- Licensing — shared license folder after deploy
- Validation notes
- Troubleshooting — add-ins / TDA

QUANTO USER GUIDE

Licensing

21-day grace, 30-day old data, instrument serial .lic, network LicenseFolder, prerelease expiry.

Instrument serial-number licensing for Quanto. Values below come from `LicenseChecker` in the product code. This is **not** 21 CFR Part 11 electronic records or signatures — see Validation notes.

How MayProcess works

A sample may be processed when **any** of these is true:

Rule	Constant / meaning
Installation grace	`InstallationGraceDays = 21` from first use (`Settings.json` `FirstUseDate`)
Old data	`DataFileFreeLicenseAfterDays = 30` — acquisition older than 30 days is free
Serial license	Instrument serial appears in a Quanto `*.lic` file in the license folder

`MayProcess` is used for Analyze-style work. Saving has a stricter rule after grace ends (below).

Instrument serial licenses

- Licenses are per **instrument serial number** (read from the sample / data file).
- Serials are listed in `*.lic` files under the license folder.
- Unlicensed serials are still **highlighted** in the UI even during grace or for old data, so you can see what will need a license later.
- Domain / user licenses (when present in the license lists) can also authorize processing; site setups normally use instrument `.lic` files.

License folder

Item	Value
Default folder	`C:\Users\Public\Documents\Quanto\License` (`Paths.DefaultLicenseFolder`)
Override	`LicenseFolder` in `%LOCALAPPDATA%\Quanto\Settings.json`
Network	Point `LicenseFolder` at a shared folder so every PC reads the same `*.lic` files

Create the folder if it does not exist. Analysts need **read**; the person who drops new licenses needs **write**. Details: Installation and folders.

Save blocked by missing license

After the 21-day grace ends, **save is refused** when the batch still has **recent** samples (acquired within 30 days) whose serials are not licensed. Grace and data older than 30 days still allow saving.

The app can build a license-request zip on the desktop for email to the vendor (see the in-app missing-license notice).

Shift during splash

Hold **Shift** while the splash is visible to recycle settings, layouts, and/or licenses (items go to the Recycle Bin). Useful after copying a wrong license folder or resetting a demo PC. Same gesture is documented under Workspace and installation.

Alpha / prerelease expiry (separate from instrument license)

Prerelease builds show a splash expiry from `SplashIdentity`:

Item	Value
`ExpirationDaysAfterBuild`	**60** (build date + 60 days)
Message	"This is a prerelease. This copy expires on …"

That calendar lock is **independent** of instrument serial licensing. A valid `.lic` does not extend a prerelease binary past its build+60 expiry; renewing the build does not replace serial licenses.

What this guide does not claim

- No Part 11 audit trail, e-signature, or electronic-record controls are documented here — none were found as product features for that purpose. See Validation notes.

Validation notes

No Part 11 / audit / e-sign in codebase; controlled install, licenses, methods, exports.

Honest scope for QA / CSV / IT: what Quanto does **not** claim, and what labs can still control.

What was not found in the codebase

Quanto does **not** implement 21 CFR Part 11-style **audit trail**, **electronic signature**, or dedicated electronic-record controls as product features. Do not document or market the desktop app as Part 11 compliant on the basis of this release.

Instrument licensing (Licensing) and prerelease expiry are **not** substitutes for Part 11 controls.

What labs can still do

Practical controls that do not require inventing missing features:

Control	How
Controlled install	Fixed path `C:\Program Files\Quanto`; silent deploy (Silent deploy); pinned build
License folder	Shared `LicenseFolder` / Public Documents license path; restricted write
Method packages	Versioned `.qtm` on a controlled share; Replace vs Append discipline
Batch packages	Study folder + `QuantoResults\*.qtb` next to raw data; backup policy
Export evidence	Excel / PDF report export with baked plots (Reports)
Permissions	NTFS on Program Files, license folder, study shares (Installation)
Reset / support	Shift+splash recycle; Recovery zips under Local AppData

Prerelease expiry

Alpha / prerelease binaries expire **60 days after build** (`SplashIdentity.ExpirationDaysAfterBuild`). That is separate from instrument serial licenses. Plan upgrades before expiry; do not treat expiry as a validation audit feature.

Related

- Licensing
- Installation and folders
- Compatibility

QUANTO USER GUIDE

Hardware and performance

CPU, RAM, SSD, network; measured Analyze times; PC and file-server sizing.

This guide is for IT and purchasing. It explains what the Analyze pipeline actually does, which hardware limits each step, typical times on a **20-logical-processor** NVMe laptop, and how to size workstations, laptops, and file servers.

Quanto is a **local desktop** process. There is no Quanto application server. A "server" in this document is a **file server** (or instrument PC) that holds raw data, methods, and finished `.qtb` files.

The **reference times** in this guide are from a measured Analyze on the 20-thread laptop (Quanto 1.0.0.8, 2026-09-05). Other batch sizes are scaled from that run. Logbook scopes to copy when you bench a new PC: `Load from data provider`, `Integrate`, `Identify`, and `Calibrate, quantitate and outliers`.

Reference laptop (20 threads)

Quanto sets `MaxDegreeOfParallelism = logical processors − 2` (default **on**). On a 20-thread PC that is **18 workers**. Two threads stay free for the UI.

Resource	What "20-core laptop" usually means	Why it matters
CPU	20 **logical** processors (often 10–14 physical cores with hyper-threading, or 20 threads on a recent Intel / AMD mobile chip)	Extract and integrate scale across samples. Calibrate / quantitate do not.
RAM	32 GB is the practical floor (this **104,544-signal** batch needs it); 64 GB if you hold several such batches or much wider windows	The whole batch's chromatograms stay in memory.
System disk	NVMe SSD, about **3–7 GB/s** sequential, **200k–500k** random 4K IOPS	Extract is random reads of vendor files. Save/open `.qtb` is sequential zip I/O.
Network	1 Gbit Ethernet ≈ **110 MB/s** best case; Wi-Fi is lower and less stable	A MassHunter `.d` folder is thousands of small files. SMB latency dominates extract.

If Task Manager shows 20 logical processors, you match this column. Physical "20-core" desktop chips are faster per thread than thin laptops; the **scaling** (what gets faster when you add cores) stays the same.

What Analyze does (and what it waits on)

Analyze (full batch) runs in this order:

Step	Parallel?	Bound by	What it does
Get signals (extract)	Yes — one sample per worker (up to 18). Agilent also **prefetches** indexes (capped around 8 opens).	**Disk / network**, then CPU	Opens each `.d` / `.wiff`, reads TIC + every method channel (quant + qualifiers), optional smooth.
Integrate	Yes — samples, and compounds inside a sample. Per-thread integrator instance.	**CPU**	Peak detect + baseline on every chromatogram.
Identify	Yes (sample loop)	**CPU**	Pick peaks, qualifiers, ISTD links, responses.
Calibrate	No (per quant / bracket)	**CPU** (single thread, curve engine)	Fit each curve. Small compared with extract.
Quantitate	No (walk every result)	**CPU** (single thread, cheap math)	Inverse curve + dilution + accuracy. Almost free.
Save `.qtb`	Chromatogram buffers written in parallel; zip is one file	**Disk**	Compress tables + all chromatograms + integrations + optional report xlsx.
Open `.qtb`	Read zip, inflate chromatograms	**Disk / network**	Reload results without touching raw files.
Report export	Up to **4** STA threads	**CPU + disk**	One Excel/PDF pair per sample.

Re-Analyze after you already have signals skips extract and is usually **integrate + identify + calibrate** only.

Measured reference (20-thread laptop, local disk)

Batch: Agilent MassHunter LC-MS pesticides demo (`D:\MassHunter\Data\Demo\LCMS_Pesticides\`). Status bar after Analyze (Quanto 1.0.0.8): **99 samples**, **461 analytes**, **104,544 signals**, **16,837 peaks**, 1 method. Saved batch `2026-09-05.qtb` = **169,324 KB (~165 MB)**. Indexed `.d` data on a local drive. Parallel processing on (18 workers).

That file is a zip (Explorer may show a 7-Zip icon). On disk it is about **1.6 KB per signal** or **1.7 MB per sample**. RAM after open is several times larger because chromatograms are inflated to `double` arrays.

Use the status-bar **Signals** count for hardware sizing. **Peaks** (16,837) are detected peaks, not chromatograms. This run is ~1,056 signals/sample and ~2.3 channels per analyte per sample.

Full Analyze (`ProcessBatch` = 6.72 s)

Step (Logbook message)	Time	Share	Rate
Load from data provider (extract)	**3.23 s**	48%	33 ms/sample · **32,000 signals/s**
Integrate	**1.97 s**	29%	20 ms/sample · **53,000 signals/s**
Identify	**0.77 s**	11%	8 ms/sample
Reapply manual overrides	0.10 s	1%	
Calibrate, quantitate and outliers	**0.46 s**	7%	
Choose reported sample	0.12 s	2%	
UI grid commit (Results)	0.04 s		
ProcessBatch total	**6.72 s**	100%	**68 ms/sample · 16,000 signals/s**

Calibrate / quantitate / outliers inside that 0.46 s:

Substep	Time
Calibrate	66 ms
Recompute auto quant selection	51 ms
Quantitate	134 ms
Outliers (99 samples)	138 ms

Identify is mostly **Assign method items** (457 ms). Correlate, qualify, peak pick, co-elution, and ISTD link are tens of milliseconds each.

Open saved batch and start

Operation	Time	Notes
Start Quanto (paths + settings + add-ins + main docking)	~8 s	Add-ins 0.39 s; docking `LoadSettings` 1.7 s
Open `.qtb` (`OpenOrCreateBatch`)	**6.2 s**	Inflate **165 MB** / 104,544 chromatograms from the zip (~27 MB/s)
Calibrate + quantitate after open	1.1 s	Calibrate 638 ms, quantitate 206 ms, outliers 184 ms

Opening this batch is about as expensive as Analyze. Review PCs need a fast local disk too, not only extract PCs.

Scale from this run (local SSD, same method style)

Linear in **signal count** for a first IT estimate. Extract grows faster than linear on a slow network; integrate stays close to linear on CPU.

Workload	Signals	Extract	Integrate + identify	Cal + quant	Full Analyze
Measured — 99 × 461, 16,837 peaks	**104,544**	**3.2 s**	**2.7 s**	**0.46 s**	**6.7 s**
Same method, ~24 samples	~25k	~0.8 s	~0.7 s	< 0.2 s	~1.6 s
Same method, ~50 samples	~53k	~1.6 s	~1.4 s	~0.2 s	~3.4 s
Same method, ~200 samples	~211k	~6.5 s	~5.5 s	~0.9 s	~14 s
2× this channel count	~209k	~6.5 s	~5.5 s	~0.9 s	~14 s
Same 104,544 signals from a **1 Gbit share**	104,544	**15–50 s** (estimate)	2.7 s	0.46 s	**20–55 s**

SCIEX WIFF is often faster to extract over the network than Agilent `.d` (fewer large files). **Full-scan EIC** extract is slower than this MRM demo.

First-time Agilent index conversion (TDA) is extra and writes `MSTree2.bin` into each `.d`. Plan **tens of seconds to a few minutes per sample** the first time. This measured run was already indexed.

How each resource changes the number

CPU

- **Helps:** extract (after the file is in cache), integrate, identify, report pictures, UI charting.
- **Does not help much:** calibrate, quantitate, zip save, opening a huge `.qtb`.
- **Diminishing returns above ~16–20 threads** for extract: prefetch already caps Agilent opens, and disk queue depth saturates.
- Prefer **high per-core speed** (recent laptop/desktop, not a many-core server chip with low clocks) for integrate and the UI.
- Leave **2+ cores free** (the default `processors - 2`). Setting parallelism to "all cores" makes the window stutter.

Purchase: **16–20 threads** is the sweet spot for an analyst PC. **8 threads** is usable for small batches. **32+ threads** is only worth it if you also have local NVMe and large residue methods.

RAM

Chromatograms stay in RAM as `double` arrays (time + response, original and often a modified copy):

Points per chromatogram	Bytes per signal (orig + modified)	105k (measured)	210k	400k
500 (narrow MRM window)	~16 KB	1.7 GB	3.4 GB	6.4 GB
2 000 (typical)	~64 KB	6.7 GB	13 GB	26 GB
8 000 (wide window / scan)	~256 KB	27 GB	54 GB	100 GB

Add **1–3 GB** for the WPF UI, grids, docking, and report designer, plus TIC traces. The measured 104,544-signal batch is in the **~8–12 GB** working-set band at typical MRM point counts.

RAM	Comfortable batch
16 GB	Tight for this measured batch (105k signals). Possible if chromatograms are short and little else is open.
32 GB	Standard analyst laptop. Proven for 99 samples / 461 analytes / **104,544 signals**.
64 GB	Several batches, Key Review, 4K + report designer, or ~200k+ signals.
128 GB	Multi-batch review or very wide scan windows.

Windows will use the page file if you undersize RAM. Analyze then becomes **disk-bound** and can look "stuck." That is not a Quanto bug.

SSD vs HDD

Storage	Extract (local)	Save / open `.qtb`	Verdict
NVMe (Gen3/4), 3–7 GB/s	Baseline in the table above	Measured **165 MB** `.qtb` opened in 6.2 s	**Required** for the system disk and for active studies
SATA SSD	~1.5–2× slower extract (random IOPS)	Fine	Acceptable for a second data disk
7200 rpm HDD	5–20× slower extract	`.qtb` save can take minutes	Archive only
USB stick / SD	Do not use	Do not use	

Random **IOPS** matter more than sequential GB/s for Agilent `.d` folders. A cheap SATA SSD still beats a fast HDD.

Put **Windows + Quanto + the active study** on NVMe. Cold archive can be HDD or NAS.

Network

Extract reads **many small files** (MassHunter `AcqData`, ChemStation). SMB latency per file adds up.

Link	Realistic extract vs local NVMe	When to use it
Local NVMe	1×	Daily Analyze
10 Gbit Ethernet, SSD/NVMe NAS, SMB3	1.2–2×	Shared study disk if the NAS is fast and antivirus is excluded
1 Gbit Ethernet	**5–15**× slower extract	OK for opening a finished `.qtb` or loading a `.qtm`. Poor for first Analyze of a large `.d` sequence
Wi-Fi 6	Unstable; often worse than 1 Gbit	Demo only. Copy the sequence local first
VPN / WAN	Often **30–100**×	Do not extract. Copy data, or Analyze on a PC next to the files

Good on a share: `.qtm` methods, `.lic` licenses, finished `.qtb` for review, `QuantoReports`.
Bad on a slow share: first Get signals from Agilent `.d` trees, TDA conversion (it writes back into the sample folder).

Practical rule: **Analyze next to the data**. Copy the sequence to the laptop NVMe, or sit at the instrument PC. After save, copy the `.qtb` to the LIMS share.

Antivirus on the NAS or the endpoint that scans every `.bin` in `AcqData` can be as slow as a 1 Gbit link. See exclusions.

GPU and display

Quanto does not use the GPU for extract, integrate, or quantitate. An **integrated GPU** is enough. A discrete GPU only helps if you use many 4K chromatogram plots or Remote Desktop with GPU encoding.

Two 4K monitors increase RAM a little (WPF visuals) but do not change Analyze time.

Batch size (what to tell the lab)

Size the PC by the status-bar **Signals** count, not Peaks and not `samples × analytes`. This 461-analyte batch is **104,544 signals** and 16,837 peaks.

Class	Signals	RAM	Disk for `.qtb`	20-thread local Analyze
Small	< 25k	16 GB	~40 MB	~2 s
Standard	25k–80k	32 GB	40–130 MB	3–6 s
Measured pesticides	**104,544**	**32 GB**	**165 MB** (`2026-09-05.qtb`)	**6.7 s**
Large	80k–200k	32–64 GB	130–330 MB	6–15 s
Extreme	> 200k	64–128 GB	330 MB–1 GB	15 s+; keep data local

Scaled `.qtb` sizes use **1.6 KB/signal** from the measured file. A report workbook inside the zip adds extra megabytes.

Key Review (several AcqKeys / QntKeys / dilutions) multiplies **rows in the UI**, not necessarily extract time, if the extra keys already have signals. Memory and grid refresh grow with visible cells.

Copying this **165 MB** `.qtb` on 1 Gbit takes a few seconds. Opening it still needs ~6 s locally to inflate. A 1 GB batch on a slow share can take longer to open than Analyze on local NVMe. Review PCs should copy large batches local or use 10 Gbit.

Workstation and laptop purchase

Quanto needs **Windows 10 or 11 64-bit**, .NET 10 Desktop Runtime (x64), and a local SSD.

Role	CPU	RAM	Storage	Network	Notes
Analyst laptop (matches the reference)	16–20 threads, recent generation	**32 GB** (64 GB if you routinely exceed ~200k signals)	**1 TB NVMe** (OS + active studies)	1 Gbit or Wi-Fi 6; dock to Ethernet for copy	Measured: 99 / 461 / **104,544 signals** / 16,837 peaks; **165 MB** `.qtb`; **6.7 s** Analyze, **6.2 s** open.
Review / senior workstation	16–24 threads, high clocks	**64 GB**	**2 TB NVMe**	**1–10 Gbit**	Several batches open, report designer, Key Review.
Heavy multi-residue	20+ threads	**64–128 GB**	**2 TB+ NVMe**	**10 Gbit** to the data disk	Do not extract from Wi-Fi.
Instrument PC (if they Analyze there)	Vendor-min or better	32 GB	Fast local disk for the sequence	Same LAN as the lab	Best place to run TDA / first extract.
File server / NAS	Few cores are fine	32 GB+ cache	**SSD/NVMe** pool for hot studies; HDD for archive	**10 Gbit** recommended	SMB3. Exclude real-time AV on `.d` / `.qtb`. Not a Quanto host.
VDI / Citrix	8+ vCPU dedicated	32 GB session	**Local or cache** the study on the session host	Low latency to the host	GPU optional for UI. Never extract a `.d` tree from the user's home NAS over WAN.

Do not buy: 8 GB RAM PCs, HDD-only laptops, or a "Quanto server" VM with no GPU and data on another continent.

Nice but unused: ECC RAM, dual CPU, compute GPU, Linux (Quanto is Windows x64 only).

Server and lab network

1. **No application server.** Do not install Quanto on a headless VM expecting batch jobs. It is an interactive WPF app.

2. **File server** holds sequences, `.qtm`, `.qtb`, licenses. Prefer 10 Gbit and SSD for the live study volume.
3. **License share** (`*.lic`) can be a small, highly available folder. Analysts need read; one admin needs write. See installation folders.
4. **One writer per** `.qtb`. Do not let two PCs save the same batch file at once. Review copies are fine.
5. **Instrument → process → archive:** acquire on the instrument disk → copy to analyst NVMe (or Analyze on the instrument PC) → save `.qtb` → copy `.qtb` + reports to LIMS / NAS → archive raw data per SOP.
6. **Domain / roaming:** roam only `%LOCALAPPDATA%\Quanto` settings if you must; never roam `BatchReports` or studies.

Checklist for a new PC

- [ ] Windows 10/11 x64, current .NET 10 Desktop Runtime
- [ ] 32 GB RAM minimum (64 GB if status-bar Signals regularly exceeds ~200k)
- [ ] NVMe system disk; active studies on that disk or another SSD
- [ ] 1 Gbit Ethernet at the bench (do not rely on Wi-Fi for extract)
- [ ] Antivirus exclusions for `C:\Program Files\Quanto` and study data
- [ ] Write access to Agilent `.d` folders if conversion may run here
- [ ] MassHunter Quant / `TDAConverterConsole.exe` only if instruments do not already write `MSTree2.bin`
- [ ] License folder reachable (Public Documents or a share)
- [ ] Confirm Task Manager → 16+ logical processors if you promised the times in this guide

Related

- Installation and folders
- Analyst workflow: Set up an analysis

QUANTO USER GUIDE

Compatibility

Single/triple quad, SIM/MRM/SCAN, MassHunter / ChemStation / SCIEX data and methods.

What Quanto can read today: instruments, acquisition modes, raw data, and vendor methods. Formats land through **add-ins** under `AddIns\` (see Installation and folders and Creating add-ins).

Platform

Item	Requirement
OS	Windows 10 or 11, **64-bit only**
Runtime	.NET 10 Desktop Runtime (x64), unless your installer bundles it
Host	WPF desktop (`Quanto.WPF.exe`); not a service, not Linux

Sizing and Analyze performance: Hardware and performance.

Instruments and acquisition modes

Quanto is built for **GC/LC–MS quantitative** work on **single quadrupole** and **triple quadrupole (QQQ)** data.

Mode	Meaning in Quanto	Typical instrument
SCAN	Full-scan MS1 extracted-ion chromatogram (EIC)	Single quad GC-MS / MS1 scan
SIM	Selected-ion monitoring	Single quad
MRM / MS-MS	Multiple-reaction monitoring (precursor → product)	Triple quad (QQQ)
Combinations	Mixed modes in one sample (e.g. SCAN+SIM)	Supported where the vendor file exposes them

Internally, chromatogram extraction uses `ChromatogramAcquisitionMode` : `Auto` , `Scan` , `Sim` , `MsMs` . **Auto** detects from the file and tries alternates if a channel comes back empty. TIC for a mode is that mode's primary TIC — not a sum of mixed modes.

MassHunter `.D` : detector classifies SingleQuadrupole vs QQQ / MRM / SIM / SCAN from the embedded acquisition method and scan types.

ChemStation `.D` : modes are SCAN, SIM, or **ScanAndSim** (`data.ms` / `dataSim.ms`). SIM windows can come from classic `acqmeth.txt` .

SCIEX WIFF: MRM transitions from the WIFF (Clearcore2 / SCIEX APIs).

Raw data (data providers)

Vendor	Add-in (class)	What you point at	Notes
Agilent MassHunter	`AgilentMassHunterDataProvider`	`.D` sample folders	Reads MassHunter data; may use Agilent **TDA indexed-data** conversion (`TDAConverterConsole.exe`) when indexed data is required for chromatogram access
Agilent ChemStation / MSD	`AgilentChemStationDataProvider`	`.D` sample folders	**Reads ChemStation data natively** via Agilent MSDChem (`MSDChemDataFile`) — **no TDA / format conversion step**
SCIEX	`SciexWiff`	`.wiff` files	Reads SCIEX WIFF through Clearcore2 / SCIEX vendor DLLs

All three also implement sample discovery (`FindSamples`) and fill common bottle metadata when present (name, acquired time, instrument, position, dilution, dose/level hints).

Vendor native DLLs ship under:

- `AddIns\VendorSupportFiles\Agilent\`
- `AddIns\VendorSupportFiles\Sciex\`

Method import (method importers)

Source	Add-in (class)	What you point at	What comes in
MassHunter quant method	`AgilentMhMethod`	`.m` method folder (`…\DaMethod\Quant\quantitative.xml`) or `.quantmethod.xml`	Compounds, ISTD flags, transitions, expected RT, curve hints
MassHunter method **from data**	`AgilentMassHunterDataMethod`	`.D` data folder	Method inferred from acquisition / MRM content in the data file
ChemStation quant method	`AgilentCsMethod`	`.M` method folder containing `qdb.mth`	Compounds from ChemStation QDB
SCIEX method **from data**	`SciexWiffMethod`	`.wiff` file	MRM compounds / transitions built from the WIFF (standalone `.msm` is **not** supported with the current minimal SCIEX DLL set)
Quanto package	(built-in host path)	`.qtm`	Full Quanto method (analytes, quants, doses, ISTD links, types, optional stored calibration and reports)

Import UX: **Method** → **Load / Import** — **Replace** or **Append** (Append uses a new **AcqKey** for multi-instrument / Key Review). Details: Reports, import, and export.

Vendor methods do **not** set Quanto's Calibration mode; new work defaults to Initial calibration unless you change it.

Peak integration (integrators)

Integrator	Role
Quanto	Shipped high-performance peak detector (default on channels)
Slice	Placeholder / alternate integrator slot (extensibility sample)

Per-channel integrator name and settings live on the method; custom integrators are add-ins (see Creating add-ins).

Capability checklist

Capability	Status
Single Quad data (SCAN / SIM)	Supported (MassHunter + ChemStation)
Triple Quad / QQQ data (MRM)	Supported (MassHunter + SCIEX; ChemStation as exposed by MSDChem)
SCAN + SIM combinations	Supported where the file layout exposes both (e.g. ChemStation ScanAndSim)
Import MassHunter method	Yes (`.m` / `.quantmethod.xml`)
Import MassHunter method from data file	Yes (from `.D`)
Import ChemStation method	Yes (`.M` + `qdb.mth`)
Import SCIEX method from data file	Yes (from `.wiff`)
Read MassHunter data	Yes
Read ChemStation data without conversion	Yes (native MSDChem)
Read SCIEX data	Yes (`.wiff`)

Related

- Analyst import/export workflow: Reports, import, and export
- Install layout and permissions: Installation and folders
- Extending formats: Creating add-ins

QUANTO USER GUIDE

Agilent ChemStation / MSD

.D with data.ms / dataSim.ms, Scan/Sim/ScanAndSim, native MSDChem, .M + qdb.mth.

How Quanto reads classic ChemStation `.D` folders and imports ChemStation methods. For MassHunter `.D` (AcqData / TDA), see Compatibility and Troubleshooting.

Raw data layout

Item	Notes
Sample	`.D` folder
SCAN / primary MS	`data.ms`
SIM	`dataSim.ms`
Modes	**Scan**, **Sim**, or **ScanAndSim** when both files are present
SIM windows	Often from classic `acqmeth.txt`

Native reader: Agilent **MSDChem** (`MSDChemDataFile`) via the `AgilentChemStationDataProvider` add-in. **No TDA / format conversion step** — unlike MassHunter indexed data.

Method import

Source	What you point at	What comes in
ChemStation quant method	`.M` method folder containing `qdb.mth`	Compounds from ChemStation QDB (`AgilentCsMethod`)

Import uses Method → Load / Import — Replace or Append (Method design).

Vendor support files

Native libraries ship under:

```
C:\Program Files\Quanto\AddIns\VendorSupportFiles\Agilent\
```

Do not mix these DLLs with another Quanto build.

Related

- Compatibility
- Troubleshooting — MH vs CS provider confusion
- Creating add-ins

SCIEX WIFF

.wiff, multi-sample #n, MRM-centric extract, Clearcore2, method from WIFF.

How Quanto reads SCIEX `.wiff` files and builds MRM methods from the WIFF.

Raw data

Item	Notes
File	`.wiff` (and companion files as required by the vendor reader)
Multi-sample	Samples inside a WIFF are addressed as `#n` (sample index)
Extract style	**MRM-centric** — transitions from the WIFF

Reader add-in: `SciexWiff` using **Clearcore2** / SCIEX vendor APIs. Support DLLs live under:

`C:\Program Files\Quanto\AddIns\VendorSupportFiles\Sciex\`

Method import

Source	What you point at	What comes in
SCIEX method from data	`.wiff` file	MRM compounds / transitions (`SciexWiffMethod`)

Standalone `.msm` is **not** supported with the current minimal SCIEX DLL set — use method-from-WIFF.

Import: Method → Load / Import — Replace or Append (Method design).

Empty or odd chromatograms

1. Confirm the Quant MRM matches a transition in the WIFF.
2. Confirm the correct multi-sample `#n` was added.
3. Confirm `VendorSupportFiles\Sciex` is present for this build.

See Troubleshooting.

Related

- Compatibility
- Reports, import, and export
- Creating add-ins

QUANTO USER GUIDE

Getting started

About 15 minutes: example IFRA batch, Analyze, calibration, export report.

Open an example batch, run Analyze, check a calibration curve, and export a report. Screenshots for this path are already on the product site (including `IFRA-Allergens.png`).

Before you start

- Quanto is installed (see Installation and folders).
- You have a `.qtb` batch, **or** you download the IFRA allergens example below.

Optional: download the IFRA example batch

If you do not have a local batch yet:

1. Download IFRA-Allergens-57.zip (~615 MB). The site presents it as a ready-to-open example batch.
2. Unzip to a writable folder (local SSD preferred).
3. Open the `.qtb` from the unzipped package (exact file name inside the zip may vary — pick the batch package, not only the raw folders).
4. Keep the vendor raw folders next to the `QuantoResults` package — the batch remembers the data path.

More examples and walkthroughs: waleson.eu.

Fifteen-minute path

1. **Open the batch** — File → Open, or drop a `.qtb` onto the Quanto window / desktop shortcut.
2. **Review bottles** — Samples grid: types, doses, dilutions. Fix obvious mismatches before Analyze.
3. **Analyze** — Run Analyze so peaks, curves, and results refresh. Details: Set up an analysis.
4. **Open calibration** — Select a calibration result and open the calibration plot. Clear **Use** on a bad point and Analyze again if needed (ESTD / ISTD).
5. **Export a report** — View → Report, then export Excel / PDF for the current sample or the batch. Formula catalog: Report formulas. Workflow: Reports, import, and export.

What you should see

Step	Expected
Open `.qtb`	Samples, method tables, and chromatograms load
Analyze	Results and flags update; curves fit
Calibration	Multi-level plot for the current Quant
Export	`.xlsx` / PDF under the study `QuantoReports` folder (or your export path)

Next reads

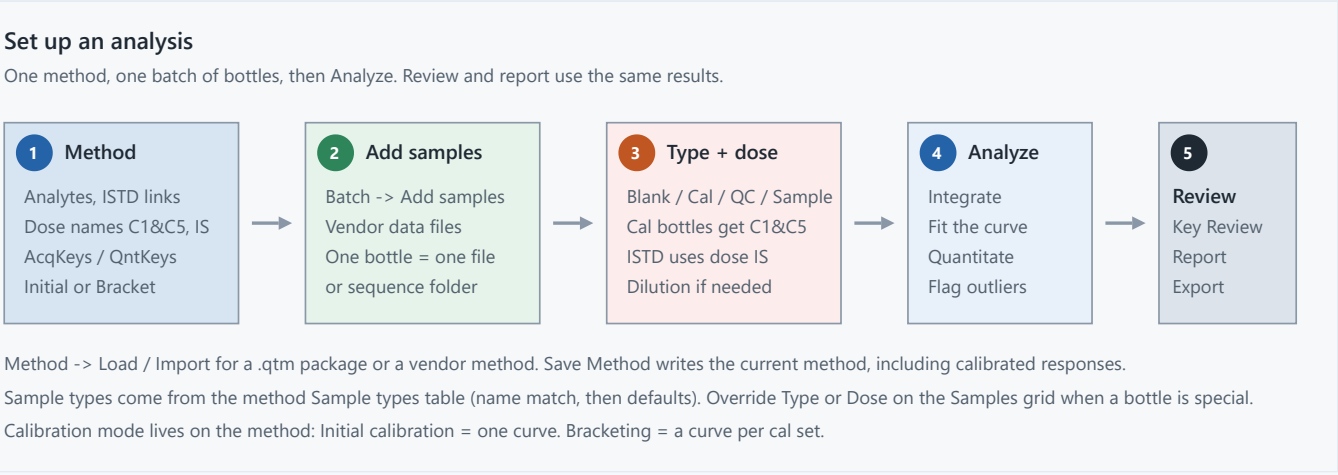
Topic	Guide
Method tables and bottle types	Set up an analysis
Method design (keys, doses, modes)	Method design
Bracketing	Bracket calibration
Multi-instrument keys	Key Review
Empty chroms / MH index / licenses	Troubleshooting
Workspace panels	Workspace and layouts

QUANTO USER GUIDE

Set up an analysis

Method, bottles, types, Analyze.

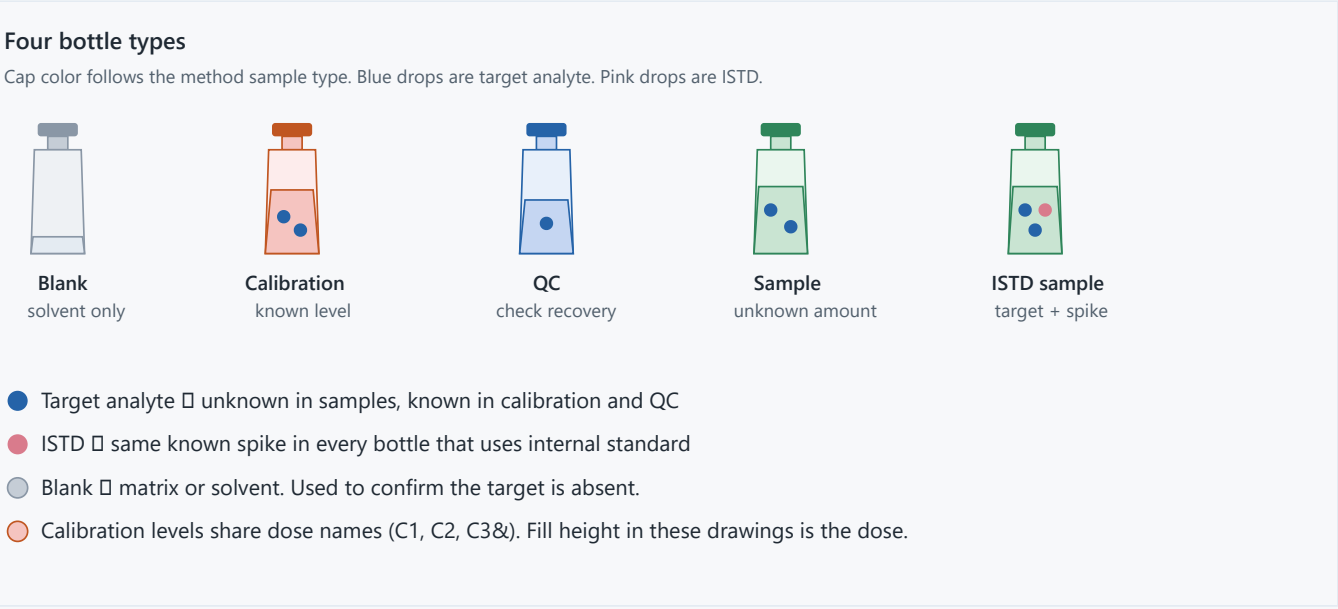
An analysis is a **method** plus a **batch of bottles**. The method says what to look for and how to build the curve. The batch is the sequence you actually ran.



Five steps: method, add samples, type and dose, analyze, review

Bottle types

Quanto has four sample types. Cap color in these drawings matches the method defaults (silver blank, coral calibration, blue QC, green sample).



Blank, calibration, QC, sample, and ISTD sample bottles

Type	What is in the bottle	What Quanto does with it
Blank	Solvent or matrix. Optional ISTD spike.	Confirms the target is absent. Not a curve point.
Calibration	Known target amount. Dose name **C1**, **C2**, â€¦	Builds the curve.
QC	Known target amount, treated as a check.	Read from the curve. Recovery / residual flags.
Sample	Unknown amount.	Read from the curve. Reported concentration.

Blue drops in the drawings are **target analyte**. Pink drops are **ISTD**. A bottle can hold several analytes; each analyte has its own curve.

Build the method

Method â†’ Load / Import

- Open a saved Quanto package (`.qtm`), or
- Import a vendor method (MassHunter, MassHunter data method, SCIEX WIFF), or
- Append another method as a new **AcqKey** (second instrument or second acquisition method).

Then check these method tables:

Table	Set this
Analytes	Name, type (Target / ISTD / Surrogate), units, extraction window
Quants	QntKey (ion / transition), curve fit, which **ISTD** Quant to use
Doses	Shared names: **C1â€¦C5** for curve levels, reserved **IS** for the ISTD spike
Quant doses	Which levels that Quant actually uses on the curve
Sample types	How file names / metadata become Blank, Calibration, QC, Sample
AcqKeys	One set per imported method or instrument
Calibration mode	**Initial calibration** (one curve) or **Bracketing** (a curve per cal set)

A Quant with no ISTD link is **ESTD**. A Quant that points at an ISTD Quant is **ISTD**. You can mix both in one method.

Load the bottles

1. **Batch → Add samples** and pick the data files (or a sequence folder).
2. Each file becomes one sample. Type and dose are matched from the method, then from the file.
3. Fix leftovers on the **Samples** grid: **Type**, **Dose**, **Dilution**, **AcqKey**, **SmpKey**.

Calibration bottles must have a dose name that exists on the analyte. Samples and blanks usually have no curve dose; ISTD bottles still use dose **IS**.

Analyze

Analyze integrates, fits each Quantâ€™s curve, quantitates, and flags. After that:

- Open the calibration plot for the current result.
- Clear **Use** on a bad cal point and Analyze again.
- Open **Key Review** when the same sample exists on several AcqKeys, QntKeys, or dilutions.

Which guide next

You are doing	Read
One instrument, no ISTD, C1â€“C5 then unknowns	ESTD initial calibration
ISTD spiked in every bottle	ISTD initial calibration
Cal, samples, cal, samples	Bracket calibration
Two instruments or several ions per analyte	Key Review
Re-run only some levels in later batches	Multi-batch dose responses
Spreadsheets, vendor files, Excel / PDF	Reports, import, and export

Grid gestures (fill, sort, freeze) are in Grids.

QUANTO USER GUIDE

Method design

AcqKey/QntKey/SmpKey, Replace vs Append, ESTD/ISTD, doses, thresholds, Initial vs Bracket.

Keys, import modes, ESTD/ISTD, doses, area thresholds, and calibration mode — the method tables you edit before Analyze.

Keys

Key	Meaning
AcqKey	Acquisition / method set. Append import creates a new AcqKey so several instruments or acquisition methods share one Quanto method.
QntKey	Quant ion / transition identity within an analyte.
SmpKey	Sample key for grouping injections (Key Review winners).

Multi-instrument pick: Key Review.

Replace vs Append (method import)

Method → Load / Import:

Mode	Effect
Replace	Swaps the current method; sample results are cleared
Append	Adds the import under a new **AcqKey**; keeps existing analytes

You can append only selected AcqKey + name keys when the dialog offers a candidate list. Details: Reports, import, and export.

ESTD vs ISTD

Setting	Meaning
ESTD	Quant has no ISTD link — external standard curve
ISTD	Quant points at an ISTD Quant — ratio / internal standard

You can mix ESTD and ISTD Quants in one method. Guides: ESTD, ISTD.

Doses

Shared dose names on the method:

Dose	Typical use
C1, **C2**, …	Calibration levels
IS	Reserved ISTD spike amount

Calibration bottles need a dose that exists on the analyte. ISTD bottles still use **IS**. Partial update of stored levels: Multi-batch dose responses.

Area thresholds

When importing integrator peaks, insignificant peaks are also filtered by a method/channel area (or height) threshold:

Setting	Default
`AreaThresholdPercent` / `DefaultPercent`	**30**
Dose reference	**Lowest** calibration concentration (or a named dose)

Threshold = calibration response × percent / 100. Channels can override with an absolute threshold instead of the general percent. See Manual integration.

Calibration mode

Mode	Behavior
Initial	One curve from the calibration set (default for new work)
Bracket	A curve per calibration bracket around samples

Vendor method import does **not** set Calibration mode — you choose it in Quanto. Guide: Bracket calibration.

Checklist before Analyze

1. Analytes / Quants / Qualifiers and extraction windows.
2. AcqKeys after any Append.
3. Doses C1… and IS; Quant doses on the curve.
4. Sample types mapping.
5. ESTD/ISTD links and QuantitateByHeight if you use height.
6. Area threshold percent / dose.
7. Calibration mode Initial vs Bracket.

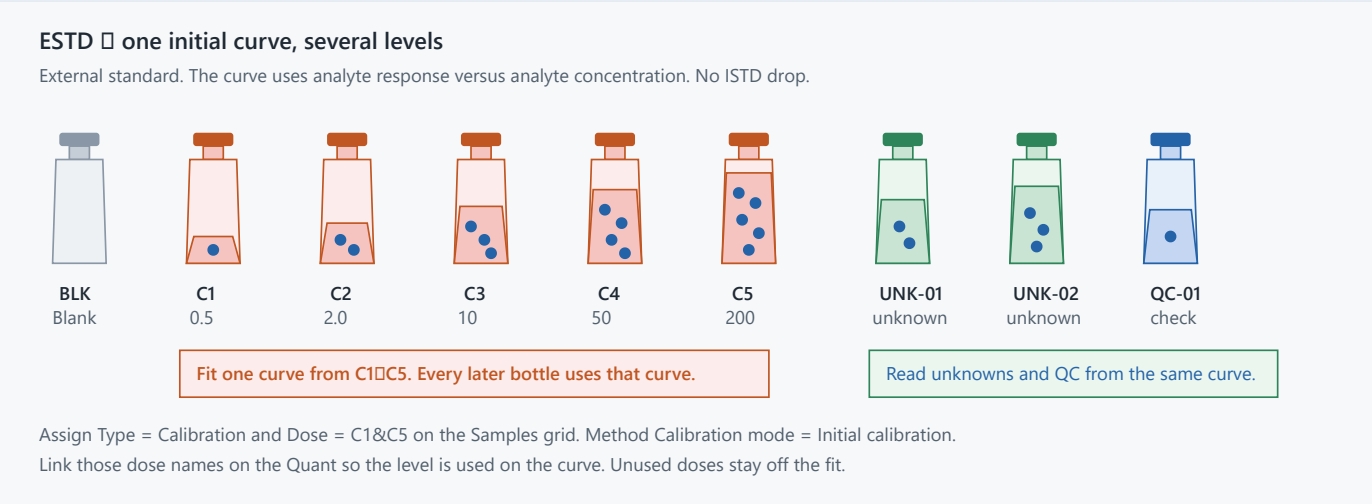
Then load bottles and Analyze (Set up an analysis).

QUANTO USER GUIDE

ESTD initial calibration

External standard, multi-level, one curve.

External standard. The target is in the calibration bottles at known levels. Samples have an unknown amount. There is **no ISTD drop**. One curve serves the whole batch.



Blank, C1–C5, unknowns, and QC

When to use it

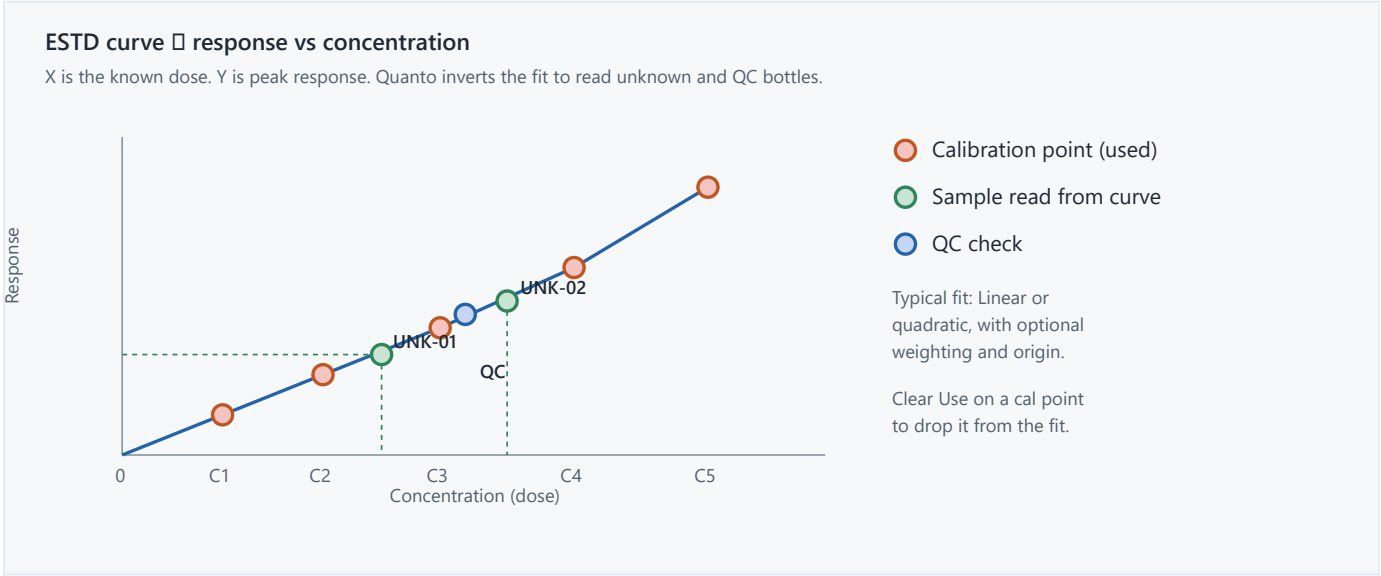
- One instrument, one acquisition method.
- You trust injection volume and recovery enough not to spike an ISTD.
- You ran a full multi-level set once (or you will reuse a saved method curve — see multi-batch).

Method **Calibration mode** = **Initial calibration**. Quant **ISTD** column empty.

Set it up

1. Import or build the method. Create dose names **C1**, **C2**, **C3**, … and type the concentrations on each analyte.
2. On the Quant, assign those doses (Quant doses). Only assigned doses go on the curve.
3. Add the sequence. Mark bottles: - Blank → Type **Blank** - Standards → Type **Calibration**, Dose **C1** … **C5** - Checks → Type **QC**, Dose of the check level - Unknowns → Type **Sample**
4. **Analyze**.

How the curve is read



Response versus concentration with unknowns read from the line

Axis	Value
X	Known dose concentration (÷ sample dilution)
Y	Peak response

Quanto fits the Quant's **Curve fit** (linear, quadratic, average response factors, power, first- or second-order ln), with the Quant's weighting and origin. It then inverts the fit for every Sample and QC.

A blank is not a curve point. Use it to confirm the target is gone, not to pin the origin, unless you chose a through-origin fit.

Multi-level rules

- Two or more distinct levels: the chosen fit is used.
- A single level: Quanto forces **average of response factors** through the origin so the one point can still be inverted.
- Clear **Use** on a point to drop it. The rest of the batch keeps the same curve (initial calibration is global).

After Analyze

- Residuals and back-calculated concentrations appear on the calibration rows.
- QC is read like a sample and flagged if it misses the method limits.
- Sample **FinalConcentration** = curve concentration × **Dilution**.

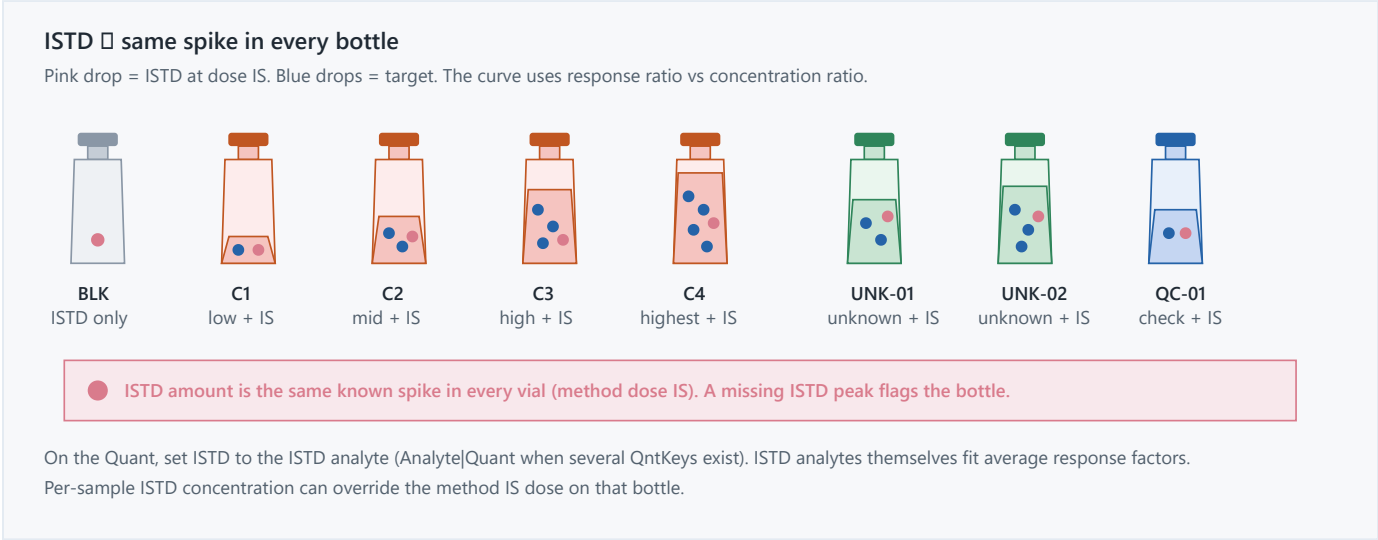
Next: add an ISTD (ISTD), split the sequence into brackets (bracketing), or keep this curve and update only some levels later (multi-batch).

QUANTO USER GUIDE

ISTD initial calibration

Internal standard, ratios, dose IS.

Internal standard. Every bottle that you intend to quantitate carries the **same known ISTD spike**. The curve is built from **ratios**, so a weak injection moves both peaks together and still lands on the line.



Blank, multi-level cals, samples, and QC, each with a pink ISTD drop

When to use it

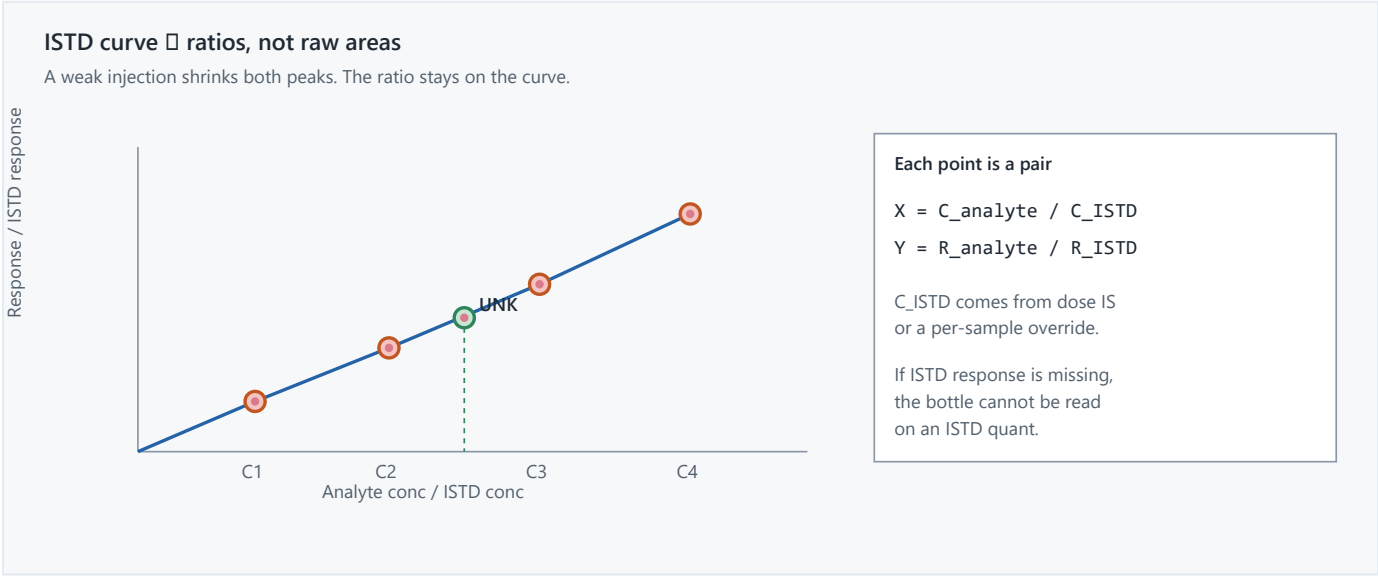
- Recovery, evaporation, or injection volume varies.
- You spike the same ISTD (or a labeled analog) into blanks, cals, QCs, and samples.
- You still want one global curve (**Initial calibration**).

Set it up

1. Create the **ISTD** analyte (type **ISTD**) and its Quant.
2. Create the target analyte (type **Target**). On its Quant, set **ISTD** to that ISTD Quant. When several QntKeys exist, the cell shows `Analyte|Quant`.
3. Keep the reserved dose name **IS**. Put the spike concentration on the ISTD analyte (and on the target if you override per bottle).
4. Create **C1…C5** (or your levels) for the **target** only. Those are the curve levels.
5. Spike ISTD into every bottle you will read. Mark cals with Dose **C1…C5**. The ISTD amount stays **IS**.
6. **Analyze**.

A blank in ISTD work usually still has the pink drop. It should show ISTD and no target.

How the curve is read



Ratio curve: analyte over ISTD on both axes

Axis	Value
X	Target concentration ÷ ISTD concentration
Y	Target response ÷ ISTD response

Quanto resolves ISTD X and Y from the linked ISTD result in the same file (or the ISTD response stored on the calibration point). If ISTD response is missing or ≤ 0, that bottle cannot be read on this Quant.

ISTD analytes themselves are fitted as **average of response factors**. They are the denominator, not a second target curve.

Overrides

- **Compound ISTD concentration** on a sample overrides the method **IS** dose for that bottle.
- Dilution divides the on-column concentrations the same way as ESTD.
- Outlier **ISTD % deviation** on the analyte flags a bottle whose ISTD response wandered.

Mixing ESTD and ISTD

One method can have ESTD quants and ISTD quants. Key Review and reports still pick per analyte. A target without an ISTD link behaves as in ESTD.

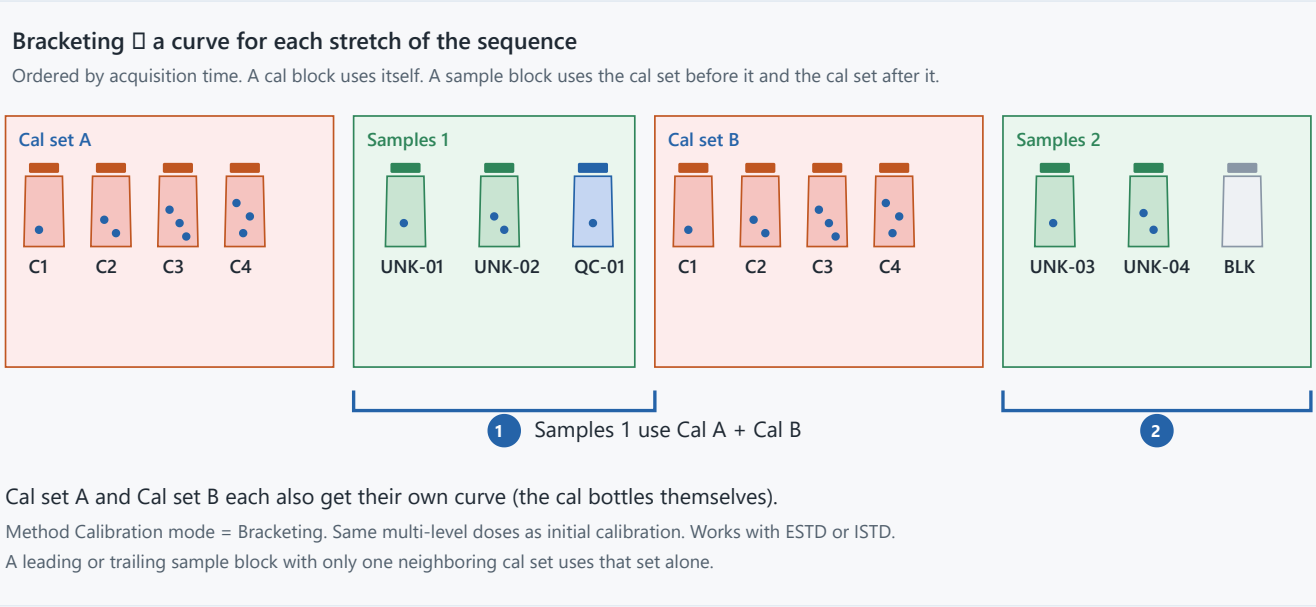
Bracketing uses the same ratios; only the *set* of cal bottles changes. See Bracket calibration.

QUANTO USER GUIDE

Bracket calibration

Cal sets around samples, per-bracket curves.

Bracketing splits the sequence by acquisition time. Each **cal set** gets its own curve. Each **stretch of samples** is read from the cal set before it and the cal set after it.



Cal set A, samples, cal set B, more samples

When to use it

- Long sequences where the instrument drifts.
- You already run opening and closing multi-level sets (and maybe mid-sequence sets).
- You want a bad cal in set B to stay off set A.

Method **Calibration mode** = **Bracketing**. Levels and ISTD links are the same as ESTD or ISTD.

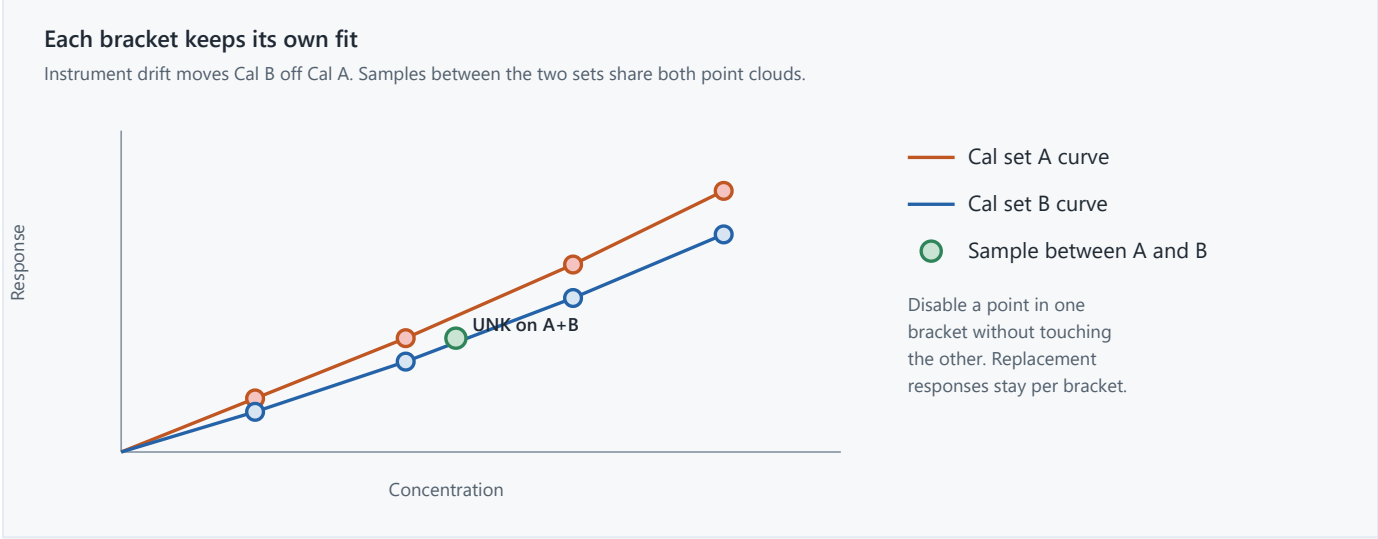
How Quanto cuts the sequence

Samples are ordered by **Acquired**. Contiguous calibration bottles form a cal set. Everything else (Sample, QC, Blank) forms a sample block.

Block	Curve points
Cal set	That set only
Sample block	Immediately preceding cal set **and** immediately following cal set
Leading samples (no cal yet)	The first cal set only
Trailing samples (no cal after)	The last cal set only

QC and blanks ride with the sample block they sit in. They do not start a new bracket.

Independent fits



Two slightly different curves from cal set A and cal set B

Each bracket stores:

- Its own **Use** flags
- Its own replacement responses
- Its own coefficients, R^2, and residuals

Turn a point off in bracket 2 and bracket 1 is unchanged. The calibration plot for a sample shows the points of **that** sample's bracket.

Set it up

1. Build the method as ESTD or ISTD, multi-level.
2. Set **Calibration mode** to **Bracketing**.
3. Add the sequence in run order (or trust **Acquired**).
4. Mark every standard Type **Calibration** and Dose **C1…Cn**. Repeat the same dose names in every cal set.
5. **Analyze**.

If a sample block has no neighboring calibrations at all, that block is not quantitated.

What bracketing does not do

- It does **not** reuse responses saved in the method. Only live batch calibrations are fitted. Partial method update is an initial-calibration feature.
- It does not average two curves into one. A sample between A and B is fitted on the **combined** A+B point cloud as one bracket.

Open the result's calibration rows to see which files belong to the current bracket.

QUANTO USER GUIDE

Key Review

Multi-instrument AcqKeys, QntKeys, SmpKey pick.

AcqKey is an acquisition set (an imported method, or an instrument). **QntKey** is one quantitation ion or transition. **SmpKey** is the reporting key that groups injections of the same physical sample.

Key Review is the matrix where those three meet: you see every AcqKey × QntKey × dilution for one SmpKey, then pick the result that reports.

Key Review one SmpKey, several instruments and ions

SmpKey SAMPLE-14 was injected on LC-QQQ and GC-MS, neat and 10x. Columns are QntKeys under dilution bands.

SmpKey SAMPLE-14						Dilution band 1x		Dilution band 10x
Group	Sum	Analyte	AcqKey	Sel	Rej	Q1	Q2	Q1 (10x)
Stimulants	12.4	Caffeine	LC-QQQ	ï		12.4 Q 195->138	11.9 Q 195->110	□
		Caffeine	GC-MS			13.1 above curve	□	1.28 diluted
Xanthines	3.1	Theobromine	LC-QQQ	ï		3.10 Q 181->138	fail ion ratio	□

Chosen result for this analyte on the SmpKey (reports use this)

Failed qualifier / flag still visible, not auto-picked

AcqKey = acquisition set (instrument or imported method). Q1 / Q2 = QntKey.

Tick Reject to drop a row from Group sum. Auto-pick prefers a passing neat result over a diluted or above-curve one.
The side Amount grid lists one concentration per SumGroup for the selected sample key.

Key Review matrix for SAMPLE-14 across LC-QQQ and GC-MS

Acquisition sets

AcqKey and QntKey how many methods fit in one batch

An AcqKey is an imported acquisition set. A QntKey is one quantitation ion / transition on that analyte.

AcqKey LC-QQQ

Imported MassHunter method

- Caffeine
  Q1 195 -> 138
  Q2 195 -> 110
- Theobromine
  Q1 181 -> 138
  ISTD caffeine-d9

AcqKey GC-MS

Appended second method

- Caffeine
  Q1 m/z 194
  own curve and ISTD
- Sample file
  AcqKey = GC-MS
  matches this slice

Same analyte name on two AcqKeys is allowed. Key Review stacks those rows. Navigator AcqKey checkboxes hide a whole instrument.
ISTD links stay inside an AcqKey unless you point a Quant at another Analyte|Quant explicitly.

Two AcqKeys, each with its own analytes, QntKeys, and ISTD

Key	Meaning
AcqKey	Slice of the method. Append a second vendor method to create one. Samples carry AcqKey so they match the right slice.
QntKey	One Quant on that analyte (Q1, Q2, …). Each has its own curve and optional ISTD.
SmpKey	Manual override, or the auto key from the method extraction table. All files that share it compete for one reported result per analyte name.
Dilution band	$\log_{10}$ of Dilution (1→0, 10→1, 100→2). Columns group by band.

Navigator **AcqKey** checkboxes hide a whole instrument from every grid, including Key Review.

Open Key Review

View → **Key Review** (or the ribbon). It follows the current SmpKey.

Rows: **SumGroup** → **analyte name** → **AcqKey**.
Fixed columns: Group, Group sum, Analyte, Selected, Reject, AcqKey.
Then one super-cell per **QntKey** under each dilution band.

The side **Amount** grid lists one concentration per SumGroup (the group sum of accepted analytes).

How a cell is chosen

Auto-pick, per analyte name inside the SmpKey:

1. Skip rejected samples and results rejected for the key.
2. Prefer a result that is **not above the curve**.
3. Prefer the **lower dilution band** (neat over 10×).
4. You can override with **Selected**, or drop a row with **Reject**.

The green outline is the winner. Reports that call `Q.KeyCompound` read that winner.

Failed qualifier / ion-ratio cells stay visible so you can see why they lost.

Typical setups

Two instruments, same samples. Append the GC method as AcqKey `GC-MS`. Give both data files the same **SmpKey**. Key Review stacks LC and GC rows; pick one (or keep both in Group sum if they are different SumGroups).

Several ions, one instrument. One AcqKey, Q1 / Q2 / Q3 columns. Qualifier percent lives on the Quant; Key Review shows pass/fail in the super-cell.

Dilution series. Same SmpKey, Dilution 1 and 10. Auto-pick keeps the neat result when it is on-scale, and falls back to 10× when the neat point is above the curve.

Group sum

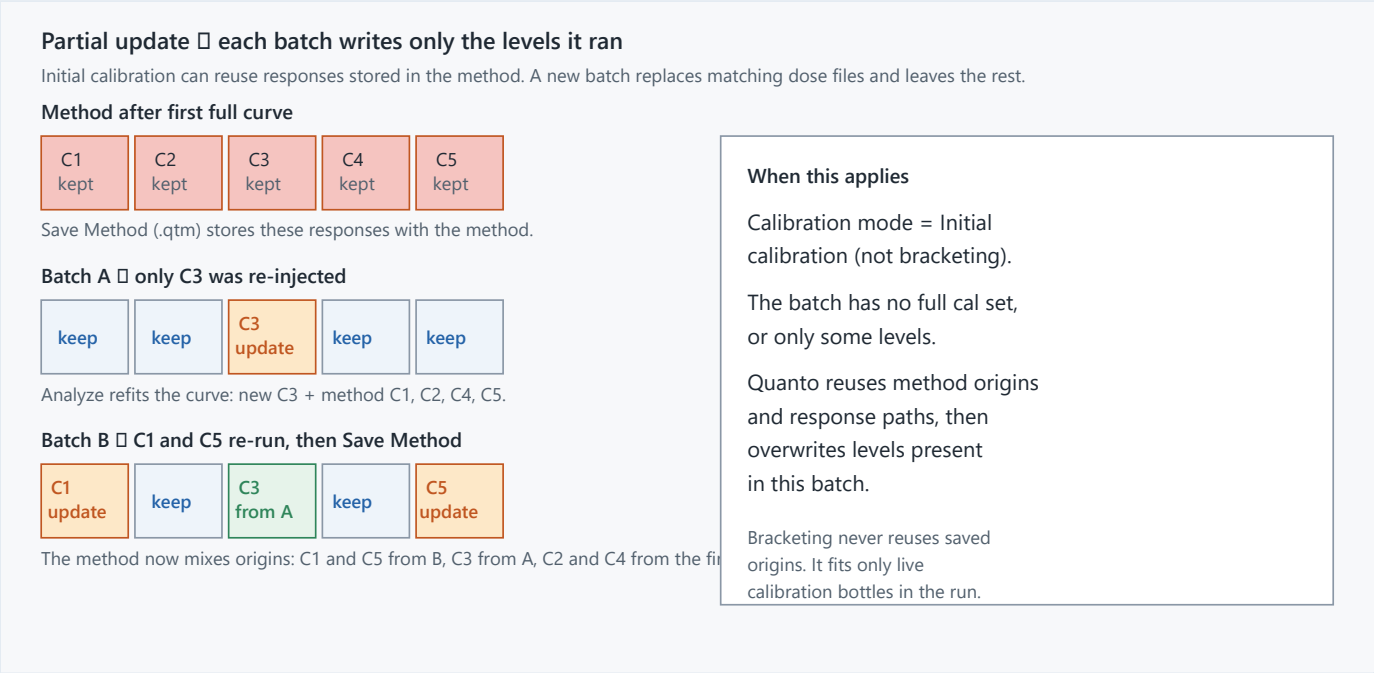
SumGroup on the AnalyteNames master (isomer family, residue group). Group sum adds accepted members. Bracketed isomer names can cover across SumGroups. Reject a row to keep it out of the sum without deleting the result.

QUANTO USER GUIDE

Multi-batch dose responses

Partial update of stored calibration levels.

After a full initial calibration you can **Save Method**. The `.qtm` keeps the curve **responses** (the Y values at C1…C5), not only the recipe. Later batches can re-run **some** levels. Quanto keeps the others.



Method levels stay; Batch A updates C3; Batch B updates C1 and C5

When to use it

- Daily work: one or two fresh standards, not a full curve.
- A level failed and you re-injected only that dose.
- Several batches should share one living method file.

This is **Initial calibration** only. Bracketing always refits from live cal bottles in that batch and never reads saved origins.

How a batch updates the method

1. First batch: full **C1…C5**, Analyze, review the curve, **Method → Save Method**.
2. Next batch: add unknowns plus only the levels you want to refresh (for example C3).
3. Analyze. Missing levels are filled from the method's stored responses and calibration-sample origins.
4. The new C3 response replaces the stored C3. C1, C2, C4, C5 stay.
5. Save Method again if this batch should become the new source for the next one.

Origins remember file path and acquisition time. A response is replaced when this batch contains that calibration path (or the same dose in the live set). Other doses are left alone.

What is stored

The method package records, per Quant:

- Dose names and concentrations
- Response values used on the curve
- Calibration-sample origins (path, acquired)
- Curve settings (fit, weight, origin, ISTD link)

It does not store unknown samples. Those stay in the `.qtb` batch.

Practical rules

- Keep dose **names** stable (C1 stays C1). A renamed dose will not match the stored point.
- If you change a concentration on the Dose table, re-run that level or accept that old responses now sit at a new X.
- A batch that contains a **complete** live cal set fits that set and does not need the stored origins.
- ISTD links still apply: a partial update of the target levels uses the ISTD response from the same file when present.

Reports and Key Review in each batch see the curve that batch actually fitted (mix of live + inherited points).

QUANTO USER GUIDE

Manual integration

PeakValues boundaries, host metrics, 2% filter, area threshold, override/restore.

How peaks move from the integrator add-in into area, height, and response — and how manual overrides are stored and restored.

What the integrator returns

An integrator add-in (`IIntegrator.Integrate`) returns only boundary coordinates on each `PeakValues` object:

Field	Meaning
`BegX`	Peak start time (X)
`EndX`	Peak end time (X)
`BegB`	Baseline intensity at start
`EndB`	Baseline intensity at end

It does **not** return Area, Height, CenterX, or width metrics. Those are computed by the host.

Host metrics: IPeakMetricsCalculator

`ImportPeakValuesService` and `PeakIntegrationService` call `IPeakMetricsCalculator` / `PeakMetricsCalculator` with the signal arrays and the four boundaries. Derived metrics include:

Metric	Role
Area	Trapezoidal area above the linear baseline
Height	Apex height above baseline
CenterX	Apex time (stored as integration Time)
FwAt50	Full width at 50% height
FwAt5	Full width at 5% height
Symmetry	Peak symmetry

Import path (automatic Analyze)

1. Integrator returns `PeakValues` (boundaries only).
2. `ImportPeakValuesService.ImportPeakValues` computes metrics, keeps peaks whose apex falls in the analyte time window (and fused abutting companions).
3. **Relative filter:** peaks below **2% of the maximum** area/height among candidates are dropped (`maxMetric * 0.02`).

4. **Area threshold:** peaks below the method/channel threshold are dropped. Default is `DefaultPercent = 30` of the **Lowest** calibration response (or a named dose). See Method design.
5. Response for quantitation is **Height** when `QuantitateByHeight` is set on the analyte; otherwise **Area**.

Manual override and restore

`PeakIntegrationService` owns live edits:

Action	Behavior
Manual boundary change	Metrics recalculated from new `BegX` / `EndX` / `BegB` / `EndB`; original automated integration retained for restore
Zero peak	Collapses boundaries to a point and re-integrates; recorded as manual
Restore original	`RestoreOriginalIntegration` puts the automated integration back (or deletes the peak if none)

Reports can distinguish current vs automated vs manual via `Peak.Area`, `Peak.AutoInt.*`, and `Peak.ManInt.*` — see Report formulas and Reports, import, and export.

Related

- Integrator add-ins: Creating add-ins
- Empty chroms / wrong provider: Troubleshooting

QUANTO USER GUIDE

Reports, import, and export

Vendor / .qtm in, Q. reports, Excel / PDF / method out.

Three doors around the same batch: bring methods and bottles in, build a spreadsheet that follows the current sample, write Excel, PDF, and method packages out.

Import, reports, and export panels

Import

Methods

Method → Load / Import is one dialog.

Source	What you get
`.qtm`	Full Quanto method: analytes, quants, doses, ISTD links, sample types, stored calibration responses, optional report workbooks
MassHunter method	Compounds, ISTD flags, transitions, expected RT
MassHunter data method	Method inferred from a `.D` data folder
ChemStation method	Compounds from `.M` / `qdb.mth`
SCIEX WIFF method	MRM compounds from the WIFF

Replace swaps the current method (sample results are cleared). **Append** adds the import under a new **AcqKey** and keeps existing analytes. That is the start of Key Review across instruments.

You can append only selected AcqKey + name keys when the dialog offers a candidate list.

Samples

Batch → Add samples

- Agilent ChemStation / MassHunter `.D` folders
- SCIEX WIFF
- Sequence or folder pick; files already in the batch are skipped

The data provider fills name, acquired time, instrument, position, dilution when the file has them. **Sample types** and **dose names** are then matched from the method. Override on the Samples grid when the file label is wrong.

What import does not invent

Vendor methods do not set Quanto **Calibration mode**. New `.qtm` packages default to **Initial calibration** unless you change it. ISTD links are imported when the vendor method has them; ESTD vs ISTD is still a Quant setting you can edit.

Reports

View → Report opens the workbook for the current method. The live sheet follows **current sample** and **current compound** (the same selection other views publish).

What a report can show

Need	Function / source
Batch header	`Q.Batch("Name")`, method, counts, last saved
This bottle	`Q.Sample("SampleName")`, Type, Dil, SmpKey, AcqKey, acquired
This analyte in this bottle	`Q.Compound("Result.FinalConcentration")`, flags, RT, ISTD
Winner across injections	`Q.KeyCompound(..., Q.SmpKeys(), Q.CurrentCompound)`
All keys in batch order	`Q.SmpKeys()`
Curve levels	`Q.Level`, `Q.Levels`, `Q.AllLevels`
Lists from the method	`Q.Analytes`, `Q.Doses`, `Q.AcqKeys`, `Q.SampleTypes`, `Q.SumGroups`
Chromatogram / curve picture	`Q.VSample`, `Q.VCompound`, `Q.VCalibration`, `Q.VQualifier`
Show or hide a block	`HideIfError`, `HideIfBlank`, `HideIfFalse`, `ShowIfTrue`

Names-grid help in the designer documents every `Source.Property` (Sample, Compound, Result, Analyte, Batch, Level, KeyReview, …).

Designer

Report Designer (stock WPF + Fluent ribbon, not Fast.Wpf) edits the workbook. **Define report** names the template. **Define export** picks:

- Which sheets belong in the **batch** pack

- Which sheets belong in each **sample** file
- Optional export folder
- Optional post-export command (run once per written file)

Workbooks can be cached into the `.qtm` when you save the method, so the next batch opens the same report.

Export

Destination	How	Contents
Method `.qtm`	Method → Save Method	Tables + calibration responses + optional report workbooks
Excel + PDF	Report export, current sample	Formulas calculated, `Q.V*` plots baked to pictures
Excel + PDF (many samples)	Batch report export	Same template, one pair of files per sample, in parallel
Grid	Copy / Copy with headers / Copy as image	Whatever is selected, including Key Review super-cells

A single-sample export needs a selected sample. Batch export walks the samples you include.

Plots are converted to embedded pictures so the `.xlsx` opens without Quanto. The live report is not modified.

Typical path

1. Import or append methods (analysis setup).
2. Add samples, Analyze, review curves and Key Review.
3. Open Report, bind `Q.KeyCompound` for the official numbers.
4. Define export sheets. Export the batch.
5. Save Method if the curve (and the report) should travel to the next batch.

Report formulas

Full Q.* catalog, named refs, HideIf* / ShowIfTrue, V* plots.

Catalog of Quanto spreadsheet functions registered in Spreado `FunctionRegistry`, with short descriptions from `QuantoFunctionDescriptions`. Workflow and export: Reports, import, and export.

Named context refs (use anywhere a sample path, compound name, or sample key is expected):

Name	Meaning	Example
`Q.CurrentSample`	Current sample relative path	`=Q.Sample("SampleName", Q.CurrentSample)`
`Q.CurrentCompound`	Current compound name	`=Q.Compound("Peak.Area", Q.CurrentSample, Q.CurrentCompound)`
`Q.CurrentSmpKey`	Current sample key (Sample keys view)	`=Q.KeyCompound("Result.FinalConcentration", Q.CurrentSmpKey, Q.CurrentCompound)`

Batch, method, samples

Function	Description	Example
`Q.Batch`	Batch field (Name, DataPath, LastSaved, ...)	`=Q.Batch("Name")`
`Q.Method`	Method field (Name, CalibrationMode, AreaThresholdPercent, ...)	`=Q.Method("CalibrationMode")`
`Q.Sample`	Sample field(s) for path(s)	`=Q.Sample("SampleName", Q.CurrentSample)`
`Q.Samples`	Vertical list of sample paths (sort/filter)	`=Q.Samples("Acquired", "Type", "=", "Sample")`
`Q.SmpKeys`	Distinct sample keys in batch order	`=Q.SmpKeys()`

Method lists

Function	Description	Example
`Q.Analytes`	Analyte names in method order	`=Q.Analytes()`
`Q.Doses`	Dose names in method order	`=Q.Doses()`
`Q.AcqKeys`	Acquisition-set names	`=Q.AcqKeys()`
`Q.Quants`	Distinct QntKey values	`=Q.Quants()`
`Q.SampleTypes`	Sample-type names	`=Q.SampleTypes()`
`Q.AnalyteGroups`	Analyte-group names	`=Q.AnalyteGroups()`
`Q.SumGroups`	Sum-group names	`=Q.SumGroups()`
`Q.AnalyteTypes`	Analyte-type names	`=Q.AnalyteTypes()`
`Q.SampleGroups`	Sample-group names	`=Q.SampleGroups()`

Compounds and results

Function	Description	Example
`Q.Compound`	Compound / peak / result fields for sample × compound	`=Q.Compound("Result.FinalConcentration", Q.CurrentSample, Q.CurrentCompound)`
`Q.KeyCompound`	Chosen result within a sample-key group	`=Q.KeyCompound("Result.FinalConcentration", Q.CurrentSmpKey, Q.CurrentCompound)`
`Q.Compounds`	Compound names for sample(s), optional filter	`=Q.Compounds(, Q.CurrentSample, "Result.Concentration", ">", 0)`
`Q.AllCompounds`	Sample/compound columns across the batch	`=Q.AllCompounds(,,"Result.Concentration",">",50)`

`Q.Compound` property paths include `Peak.*`, `Peak.AutoInt.*`, `Peak.ManInt.*`, `Result.*`, `Level.*`, `Q1.*` qualifiers, and related prefixes — see names-grid help in the designer.

Calibration levels

Function	Description	Example
`Q.Level`	Level field for compound / level key	`=Q.Level("Concentration", , Q.CurrentCompound, $B$4#)`
`Q.Levels`	Level keys for sample/compound context	`=Q.Levels(, Q.CurrentSample, Q.CurrentCompound)`
`Q.AllLevels`	Level matrix or analyte/level-key lists	`=Q.AllLevels("Result.Concentration")`

Sheet helpers

Function	Description	Example
`Q.SheetName`	Worksheet name containing the formula	`=Q.SheetName()`
`Q.TemplateName`	Report template file name (no extension)	`=Q.TemplateName()`
`Q.RepeatCells`	Tile a source range	`=Q.RepeatCells(A1:B2, 2, 3)`
`Q.ByRowAndColumn`	LAMBDA over row × column vectors	`=Q.ByRowAndColumn(A1:D1, A2:A5, LAMBDA(r,c,r&"-"&c))`
`Q.VSearch`	Find text on sheets; return relative cells	`=Q.VSearch({"Results"}, {"Limit"}, {"R0C1"})`

Show / hide blocks

Place in column A to hide rows or in row 1 to hide columns. Worksheet must contain `HIDE` in **A1** to enable visibility processing. Use `--` on spilled refs so blanks become numeric zero.

Function	Description	Example
`Q.HideIfError`	Hide when values are errors	`=Q.HideIfError(--A1#)`
`Q.HideIfBlank`	Hide when values are blank	`=Q.HideIfBlank(--A1#)`
`Q.HideIfFalse`	Hide when values are FALSE	`=Q.HideIfFalse(--A1#>0)`
`Q.ShowIfTrue`	Show only when values are TRUE	`=Q.ShowIfTrue(--I5#>10)`

(Same family as noted in the reports guide: HideIfError / HideIfBlank / HideIfFalse / ShowIfTrue.)

Visual plots (baked on export)

Function	Description	Example
`Q.VSample`	Sample TIC plot in the cell	`=Q.VSample(Q.CurrentSample)`
`Q.VCompound`	Compound chromatogram	`=Q.VCompound(Q.CurrentSample, Q.CurrentCompound)`
`Q.VCalibration`	Calibration curve plot	`=Q.VCalibration(Q.CurrentSample, Q.CurrentCompound)`
`Q.VQualifier`	Qualifier overlay (0 = quant, 1 = all, 2+ = specific)	`=Q.VQualifier(Q.CurrentSample, Q.CurrentCompound, 1)`

Related

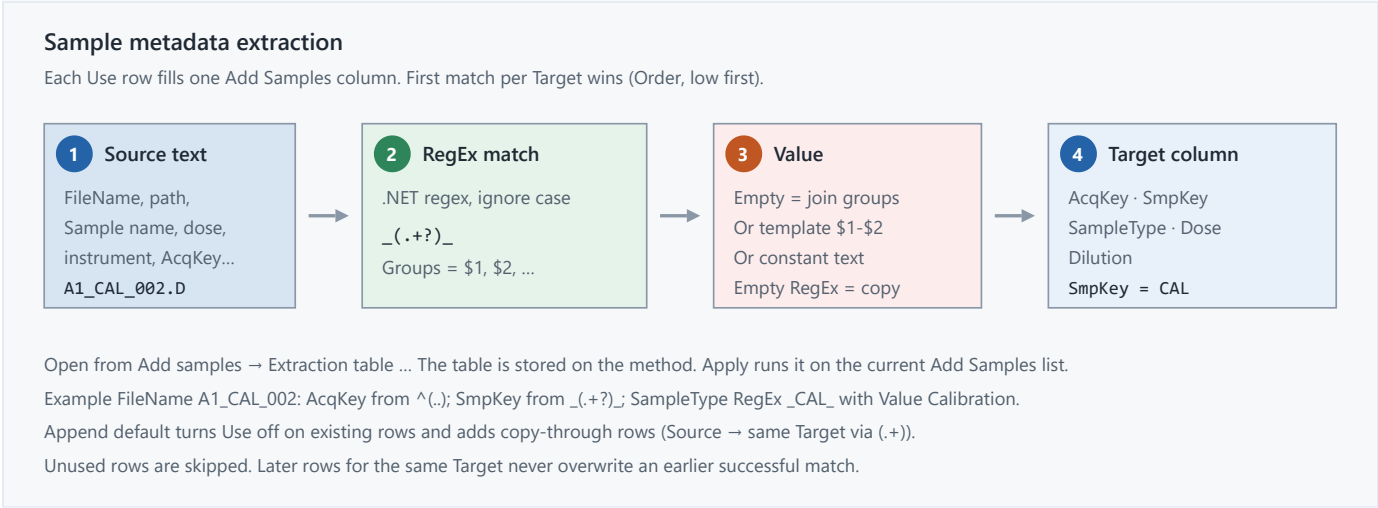
- Reports, import, and export
- Manual integration — Peak.AutoInt / ManInt
- Key Review — `Q.KeyCompound` / `Q.SmpKeys`

QUANTO USER GUIDE

Sample metadata extraction

Add Samples Extraction table - RegEx to AcqKey / Type / Dose.

When you **Add samples**, Quanto fills **AcqKey**, **SmpKey**, **SampleType**, **Dose**, and **Dilution** from a table of rules on the method. Open it with **Add samples → Extraction table ...** (title: **Sample metadata extraction**).



Source text through RegEx and Value into Target columns

What each row does

Column	Meaning
Use	Off = skip this row
Order	Low first. First successful match **per Target** wins
Target	Add Samples column to fill: AcqKey, SmpKey, SampleType, Dose, Dilution
Source	Text to read: FileName, RelativePath, Data folder, Sample name, Dose, SampleType, Dilution, instrument fields, or the method AcqKey
RegEx	.NET regular expression (ignore case). Empty = copy Source, or use **Value** as a constant
Value	Template with `$1`, `$2`, ... Empty = concatenate capturing groups

A method with no rows yet gets **copy-through** defaults (AcqKey←AcqKey, SmpKey←RelativePath, SampleType←SampleType, Dose←Dose, Dilution←Dilution) using RegEx `(.+)`.

Worked example

File names:

- `A1_CAL_002`
- `A1_A0001_001_1`
- `A1_BlankMP_1`

Target	Source	RegEx	Value	Result
AcqKey	FileName	`^..`	*(empty)*	`A1`, `A1`, `A1`
SmpKey	FileName	`_(.+?)_`	*(empty)*	`CAL`, `A0001`, `BlankMP`
SampleType	FileName	`_CAL_`	`Calibration`	Calibration on `A1_CAL_002` only

Empty **RegEx** with a filled **Value** writes that constant when the row runs. Empty **RegEx** and empty **Value** copies the Source text as-is.

Buttons

Button	Effect
Add row	Duplicate the current row (or last row) and bump Order
Append default	Turn **Use** off on existing rows; append the five copy-through defaults with Use on
Delete row	Remove the selected rows
Apply	Store the table on the method and run it on the current Add Samples list
Close	Close the dialog (layout is saved)

You need a loaded or imported method first — the table lives on the method, not only on the batch.

Tips

- Prefer **FileName** or **RelativePath** when folders encode AcqKey / instrument.
- For SampleType, match a token (`_CAL_`, `_QC_`, `_Blank_`) and set **Value** to `Calibration`, `QC`, `Blank`, or `Sample`.
- Fix leftovers on the Add Samples grid (or Samples grid after import) when a bottle is special.
- Regex help: the dialog links to the Microsoft .NET regular expression language reference.

Related

- Set up an analysis — Add samples in the full workflow
- Key Review — several AcqKeys or SmpKeys for the same bottle
- Grids — fill, sort, freeze on the extraction grid

QUANTO USER GUIDE

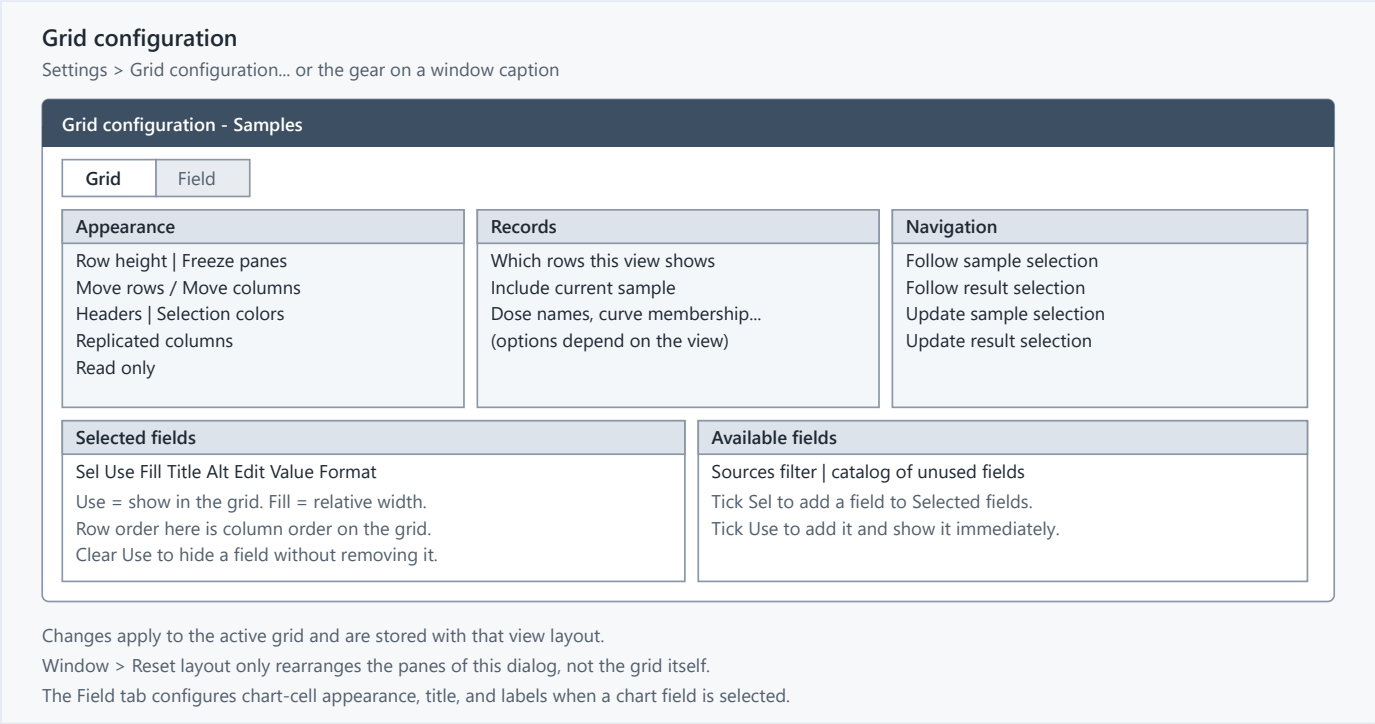
Grids: interact and configure

Column fill, sort, freeze, navigator.

This guide covers the grids you work in every day: how to resize and fill columns, set row height and how many records fit in the pane, sort, and reorder rows or columns. Settings live with the **active grid** and are stored in that view's layout.

Open Grid configuration

1. Click the grid you want to change (make it the active view).
2. Choose **Settings → Grid configuration...**
3. Or click the **gear** on a window caption (non-main windows).



Grid configuration window: Appearance, Records, Navigation, Selected fields, Available fields

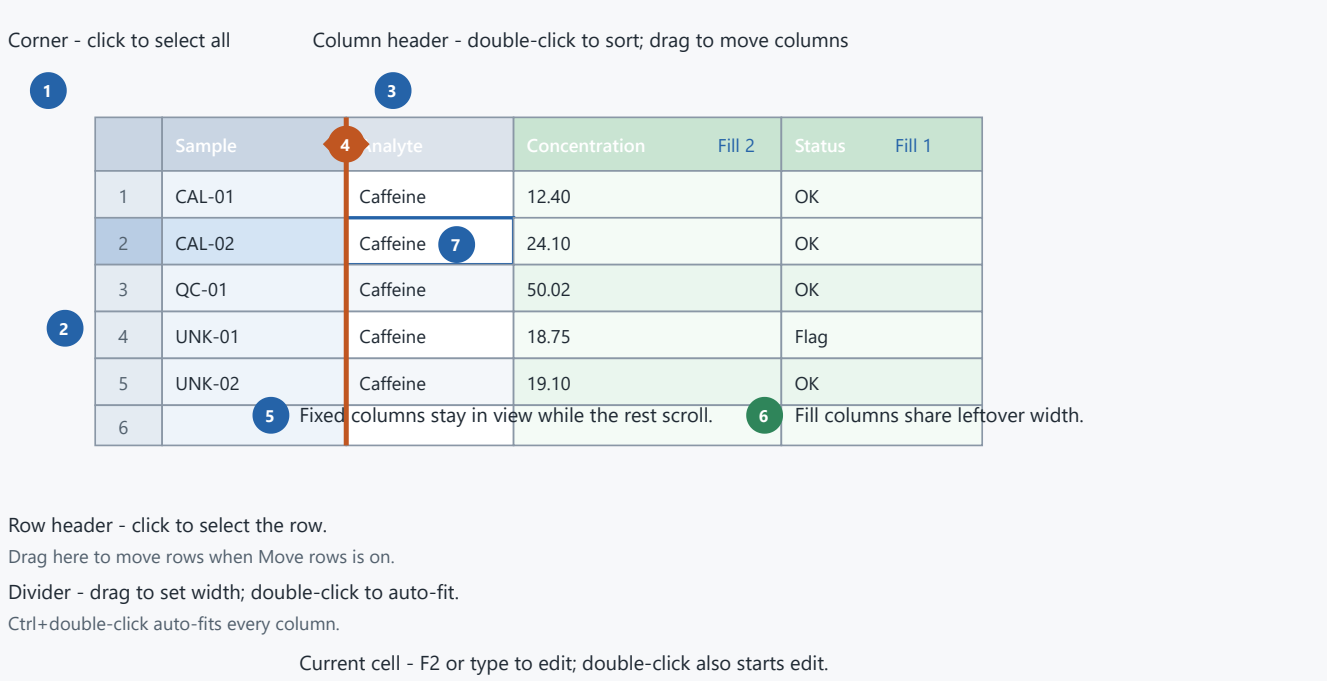
The dialog follows the grid you last clicked. Changes apply immediately and stay with that view.

Pane	What it controls
Appearance	Row height, records per pane, freeze, move rows/columns, headers, colors
Records	Extra include/filter options for this view (current sample, curve points, ...)
Navigation	Whether this grid follows or publishes sample/result selection
Selected fields	Column order, visibility, **Fill**, titles, formats
Available fields	Catalog of fields you can add

The **Field** tab is for chart cells (appearance, title, labels). **Window** → **Reset layout** only resets this dialog's panes, not the grid.

Which rows you see is also controlled by the **left navigator** explorer bar (**AcqKey**, **Sample groups**, **Analyte groups**). Those checkboxes apply to every grid and chart, not only the active view. See Navigator explorer bar.

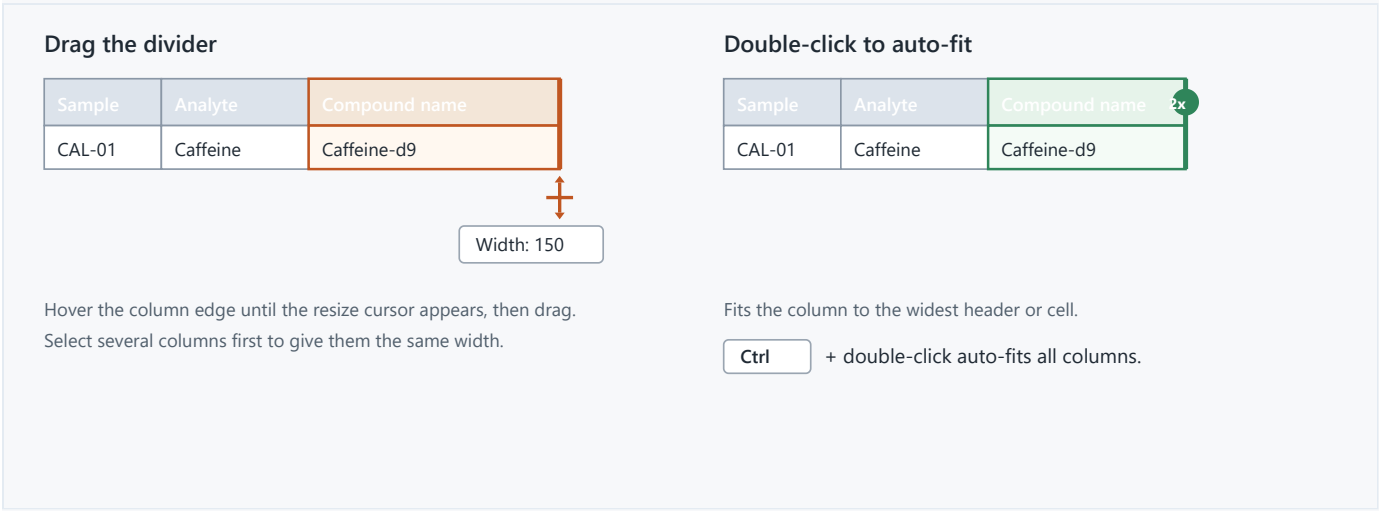
Anatomy of a grid



Labeled grid: corner, row header, column header, divider, freeze, fill, current cell

#	Part	What you do there
1	Corner	Click to select the whole grid
2	Row header	Click to select the row; drag to **move rows**
3	Column header	Double-click to **sort**; drag to **move columns**
4	Divider	Drag to resize; double-click to auto-fit
5	Fixed columns / rows	Stay in view while the rest scroll
6	Fill columns	Grow and shrink with the pane
7	Current cell	Type or **F2** to edit

Column width



Drag a column divider to resize; double-click to auto-fit

Set a width. Hover the vertical line between columns until the resize cursor appears, then drag. A live tip shows `Width: n`. You can drag in the header or in the body.

Auto-fit one column. Double-click that column's divider. The column becomes as wide as its header or widest cell.

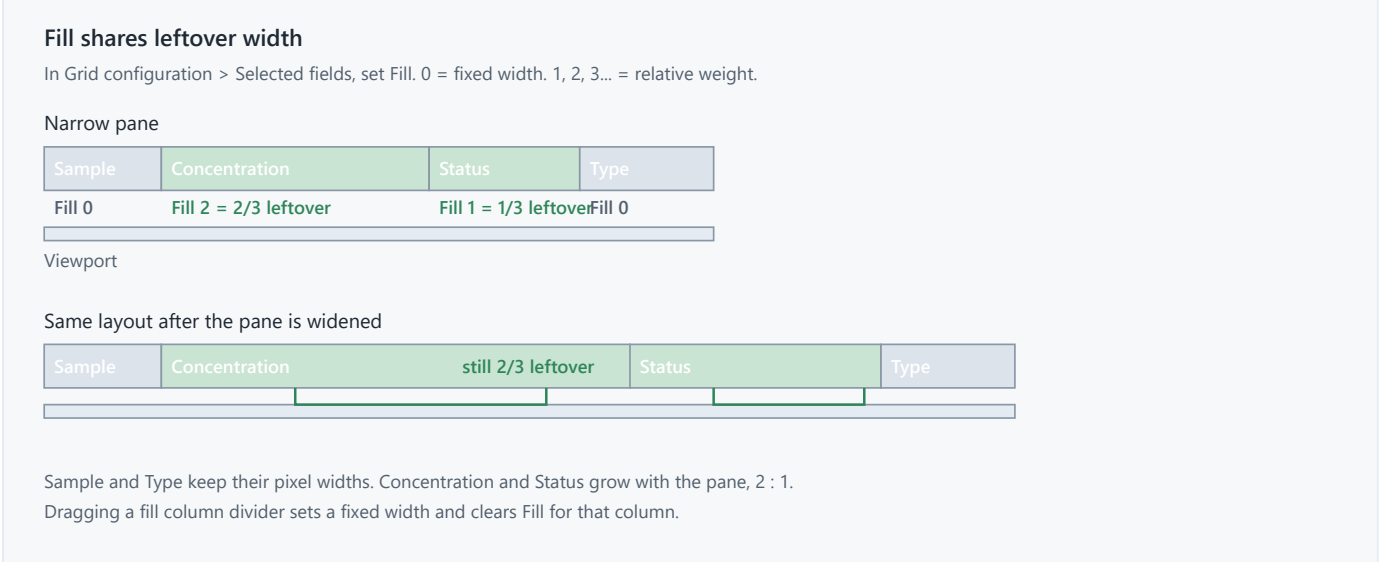
Auto-fit every column. Ctrl+double-click any column divider.

Same width on several columns. Select those columns first (click their headers; **Ctrl** or **Shift** to add more), then drag one of their dividers. All selected columns get the same pixel width.

Equalize without Fill. There is no separate "equalize" command. Multi-select + drag one divider is the way to make columns the same width.

Dragging a column that was using **Fill** turns it into a fixed-width column.

Column Fill



Fill weights: leftover pane width shared 2:1 between Concentration and Status

Fill is a relative width weight on **Selected fields**. It is the way to make columns use the leftover pane width.

Fill	Meaning
0	Fixed width (the pixel width from drag or auto-fit)
1, 2, 3…	Share leftover space in that ratio

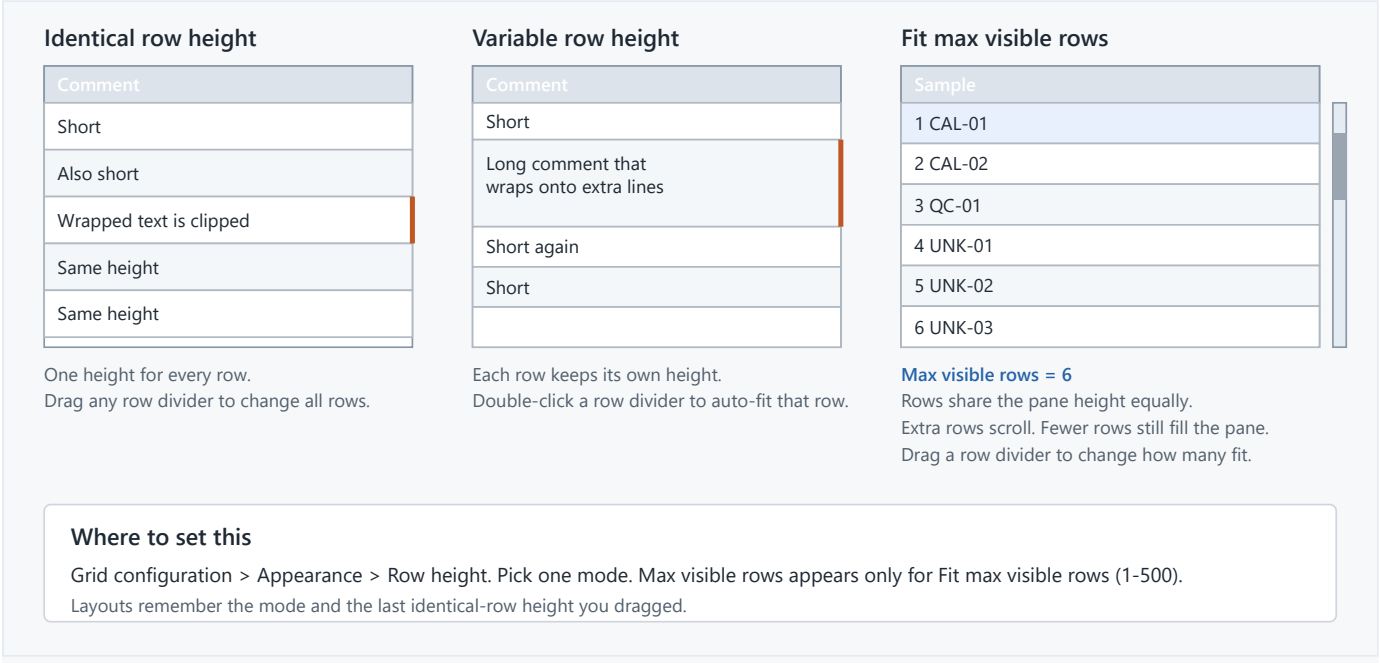
Example: Sample **Fill 0**, Concentration **Fill 2**, Status **Fill 1**, Type **Fill 0**. When you widen the pane, Sample and Type stay put. Concentration gets two thirds of the extra space, Status one third.

How to set it

1. Open **Grid configuration**.
2. In **Selected fields**, find the column.
3. Set **Fill**. `0` = fixed; any positive integer = weight.

Leave at least one column on Fill if you want the grid to use the full pane width without a large empty strip on the right.

Row heights and records per viewport



Three row height modes: identical, variable, and fit max visible rows

In **Appearance → Row height**, pick one mode:

Identical row height

Every row is the same height. Drag **any** row divider to change them all. Double-click a divider to auto-fit to the tallest content, then apply that height to every row.

Use this for dense result lists.

Variable row height

Each row keeps its own height. Drag one divider to change only that row (or all selected rows). Double-click a divider to auto-fit that row; **Ctrl+double-click** auto-fits all rows.

Use this when comments or wrapped text should make some rows taller.

Fit max visible rows (records per viewport)

Rows share the pane height so a chosen count is visible.

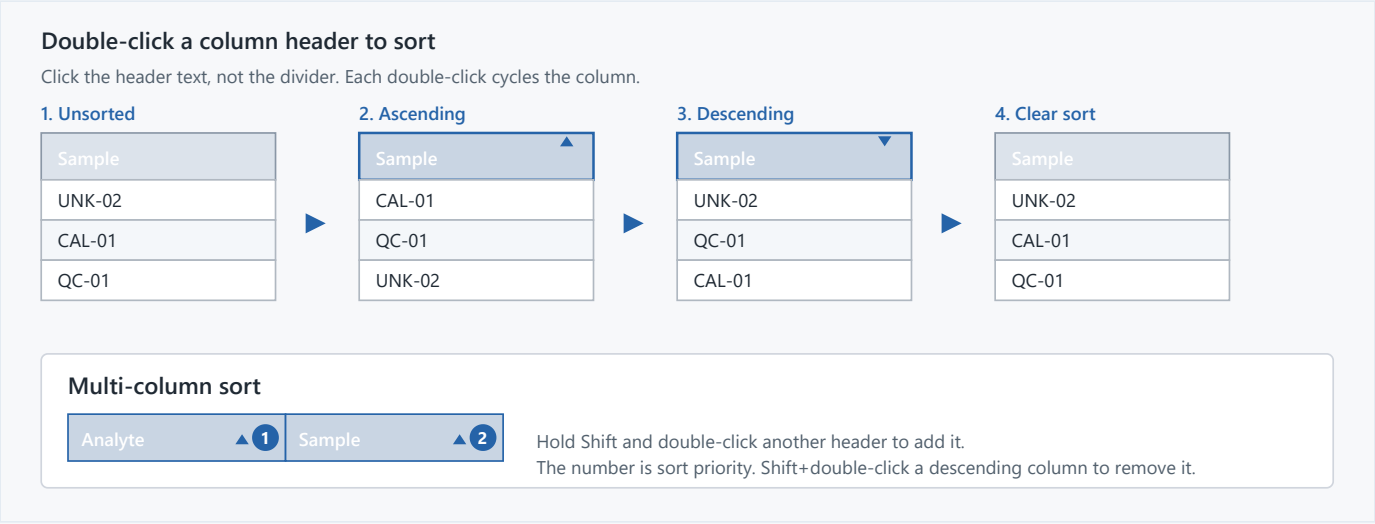
1. Select **Fit max visible rows**.
2. Set **Max visible rows** (1–500). That is how many records share the viewport.

If the grid has *more* rows than that number, you scroll. If it has *fewer*, those rows still fill the pane (no empty band under the last record).

You can also drag a row divider: the count updates from the new row height (`pane height ÷ row height`). A small drag does not collapse the grid to a single oversized row.

Typical use: details or chromatogram-in-cell views where you always want, for example, six records on screen.

Column sort



Double-click header cycles unsorted → ascending → descending → unsorted; Shift adds a second sort column

Sort one column. Double-click the **header text** (not the divider).

Cycle: unsorted → ascending (▲) → descending (▼) → unsorted.

Sort several columns. Hold **Shift** and double-click another header. A small number is the sort priority (1 then 2). Shift+double-click a descending column to drop it from the sort.

Natural order. Text sorts in natural order (`Q2` before `Q10`).

Sort blocks move. Clear the sort (cycle until the triangle is gone) before **Move rows** or **Move columns** will work.

Layouts remember the sort.

Manual order: rows and columns

Move rows

	Sample	Type
1	CAL-01	Cal
2	QC-01	QC
3	QC-02	QC
4	UNK-01	Unk
5	UNK-02	Unk

1. Turn on Move rows in Appearance > Interaction.
2. Click row headers to select whole rows (Shift / Ctrl for more).
3. Drag the selected headers. Drop on the insert line.

Move columns

	Sample	Status	Type	Conc
1	CAL-01	OK	Cal	12.4
2	QC-01	OK	QC	50.0
3	UNK-01	Flag	Unk	18.8

1. Turn on Move columns in Appearance > Interaction.
2. Click a column header so the whole column is selected.
3. Drag the header. The vertical line is the drop position.

Sort must be off. Clear any sort (double-click the sorted header until the triangle disappears) before you can move rows or columns.
You can also change column order by rearranging rows in Selected fields.

Select headers then drag to an insert line to reorder rows or columns

Move rows

1. **Appearance** → **Interaction** — turn on **Move rows**.
2. Click **row headers** so the whole rows are selected (**Shift** / **Ctrl** for more).
3. Drag the selected headers. Drop on the horizontal insert line.

Move columns

1. Turn on **Move columns**.
2. Click **column headers** so the whole columns are selected.
3. Drag the selected headers. Drop on the vertical insert line.

Another way to order columns: in **Selected fields**, the row order *is* the column order. Drag those rows (when move-rows is available on that grid) or add/remove fields to reshape the layout.

Blocked while sorted. If a triangle is showing on a header, clear the sort first.

Freeze panes

Freeze panes

Appearance > Freeze panes. Counts are body rows from the top and body columns from the left.

	Sample	Analyte	RT	Area	
1	CAL-01	Caffeine	1.02	12040	
8	UNK-06	Caffeine	1.04	9810	
9	UNK-07	Caffeine	1.03	10022	
10	UNK-08	Caffeine	1.05	8870	

Fixed columns = 1
Sample stays visible while you scroll sideways through results.

Fixed rows = 1
The first data row stays at the top while you scroll the rest.

Fixed columns stay left and fixed rows stay top while the rest of the grid scrolls

Appearance → Freeze panes

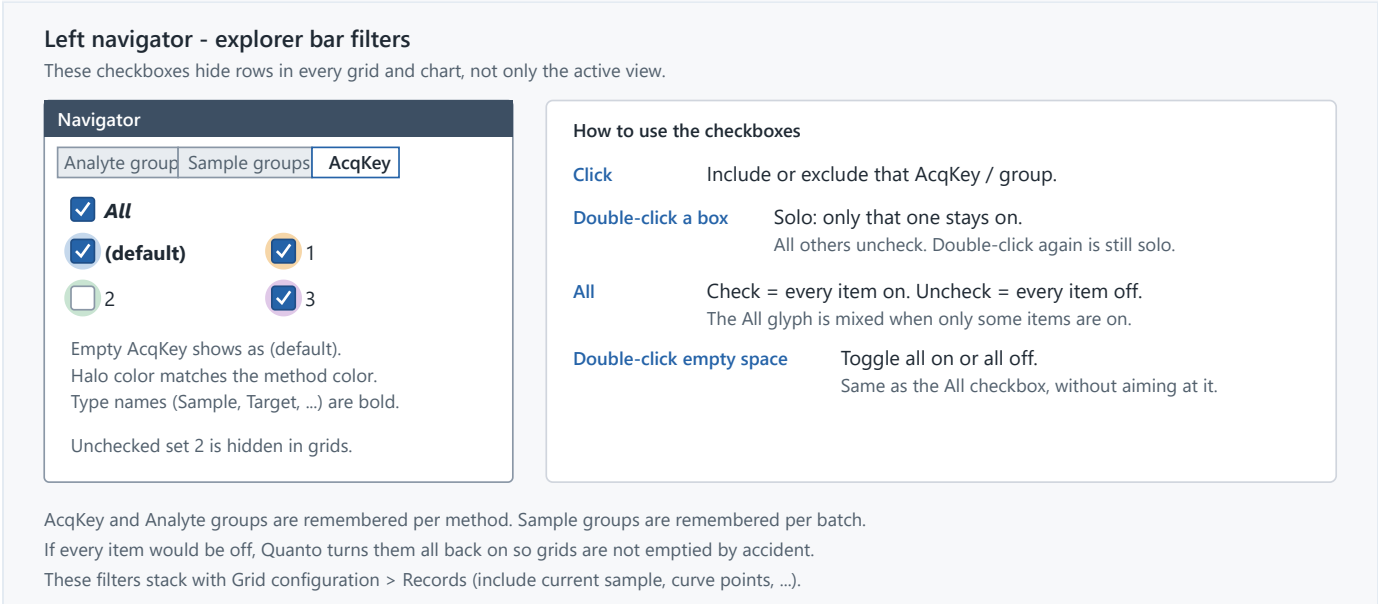
Setting	Effect
Fixed rows	That many body rows stay at the top
Fixed columns	That many body columns stay on the left

There is no “pin this column” on the header menu. Change the counts here.

Navigator explorer bar (AcqKey and groups)

The left navigator is the explorer bar. Three panes of checkboxes filter which records grids and charts show:

Pane	What it filters	Typical items
AcqKey	Samples, analytes, results, Key Review, charts	`(default)` plus named acquisition sets (`1` , `2` , …)
Sample groups	Sample lists and anything built from samples	Type categories **Sample**, **Calibration**, **QC**, **Blank**, then named groups
Analyte groups	Results, details, chromatogram cells	Type categories **Target**, **ISTD**, **Surrogate**, then named groups



AcqKey explorer bar checkboxes: All, (default), named sets; click, double-click solo, All

These are **global**. Unchecking AcqKey `2` hides that set in the Samples grid, Results, details, Key Review, and charts at once. They stack with **Grid configuration** → **Records** (include current sample, curve points, and so on).

Click a box to include or exclude that set or group.

Double-click a box to solo it: only that item stays on. Use this when you want one AcqKey or one group on screen.

All (italic, first in the list) checks or unchecks every item. The glyph is mixed when only some items are on.

Double-click empty space in the pane (not on a box) toggles all on or all off — same as **All**, without aiming at it.

Colors and labels

- A color halo around the checkbox matches the method color for that AcqKey or group.
- Type categories (`(default)` , Sample, Target, …) are **bold**.
- An empty acquisition set is listed as **(default)**.

Remembered

- **AcqKey** and **Analyte groups** are stored per method.
- **Sample groups** are stored per batch.
- New items default to on. If a stored layout would leave every item off, Quanto turns them all back on so the grid is not emptied by accident.

View → Reset navigator restores the navigator pane layout (Windows, Layouts, these filter panes). It does not reset which boxes are checked.

Fields: show, hide, title, format

Selected fields is the layout of the active grid.

Column	Meaning
Sel	Include this field in the layout
Use	Show it on the grid (clear to hide without removing)
Fill	Relative width; 0 = fixed
Title	Default header
Alt / **Alt title**	Use an alternate header (empty Alt title = blank header)
Edit	Field can be edited when the grid is not read-only
Value	Preview for the current record
Format	Number / date format
Source / **Field**	Where the value comes from

Add a field: in **Available fields**, tick **Sel** (add) or **Use** (add and show). Use the **Sources** filter if the catalog is long.

Headers on the grid itself: Appearance → Headers — row headers, column headers, row numbers, toolbar.

Select, copy, edit

Gesture	Result
Click a cell	Current cell
Shift+click or drag	Extend a range
Ctrl+click or Ctrl+drag	Extra selection areas
Click a row / column header	Select the whole row / column
Click the corner	Select all
Double-click a cell, **F2**, or type	Start edit
Space	Toggle a checkbox cell
Ctrl+click a URL	Open the link
Alt+drag in a column	Select part of the text down the column

Right-click for Cut, Copy, Copy with headers, Copy as image, Paste, Select all, Fill down (**Ctrl+D**), Increment down (**Ctrl+I**). Some views add extra items (for example Accept / Reject).

Appearance → **Interaction** → **Read only** blocks edits; selection and copy still work.

Appearance → **Selection & colors** turns selection fill, current-cell rectangle, edit rectangle, inactive-pane hiding, header tints, and alternate row colors on or off.

Keyboard

Keys	Action
Arrows	Move current cell
Ctrl+Arrow	Jump to the edge of the data
Shift+navigation	Extend the selection
Home / End	Start / end of the row
Ctrl+Home / Ctrl+End	First / last cell
Page Up / Page Down	Page by the visible rows
Tab / Shift+Tab	Next / previous column
Enter / Shift+Enter	Next / previous row
F2	Edit
Escape	Cancel edit
Ctrl+A	Select all
Ctrl+C / Ctrl+Shift+C	Copy / copy with headers
Ctrl+X / Ctrl+V	Cut / paste
Ctrl+D / Ctrl+I	Fill down / increment down
Delete	Clear the selection
Ctrl+Mouse wheel	Zoom the grid
Mouse wheel / Shift+wheel	Scroll by row / by column

While editing: **Enter** commits and moves down; **Ctrl+Enter** commits and moves up; **Tab** commits and moves across.

Navigation between views

Navigation → Record navigation

Option	Meaning
Follow sample selection	This grid reloads when another view changes the current sample
Follow result selection	Same for the current result
Update sample selection	This grid publishes its sample so other views can follow
Update result selection	Same for the result

Turn **Follow** on for a details pane that should track the sample list. Turn **Update** on for the list that should drive other views.

Other appearance options

Replicated columns (qualifier slots, dose bands, and similar):

- **Group by Source** — all fields of Q1, then all fields of Q2…
- **Group by Property** — the same property across instances, then the next property
- **Maximum count** — how many instances to expand

Settings → Appearance… (theme, Compact / Standard / Spacious density, zoom, language) sets default sizes for new layouts. It does not replace per-grid column widths you already dragged.

Cheatsheet

Goal	Do this
Open settings for this grid	**Settings → Grid configuration…** (or caption gear)
Column width	Drag the vertical divider
Auto-fit one / all columns	Double-click / **Ctrl+double-click** the divider
Columns share pane width	**Selected fields → Fill** (0 = fixed)
Same width on several columns	Select those columns, drag one divider
All rows the same height	**Identical row height**, drag any row divider
Each row its own height	**Variable row height**
N records always in view	**Fit max visible rows** + **Max visible rows**
Sort	Double-click the column header
Multi-column sort	**Shift+double-click** further headers
Reorder rows / columns	Enable **Move rows** / **Move columns**, drag selected headers (sort off)
Freeze	**Fixed rows** / **Fixed columns**
Hide a column	Clear **Use** on Selected fields
Add a column	Tick **Sel** or **Use** on Available fields
Filter by AcqKey / groups	Left navigator explorer bar checkboxes
Solo one AcqKey or group	Double-click that checkbox
Check or uncheck every filter	**All**, or double-click empty space in the pane

Analysis setup (bottles, ESTD / ISTD, bracketing, Key Review, reports) is in the user guide index.

QUANTO USER GUIDE

Workspace and layouts

Default.qtl, dockable panels, limited shortcuts, Shift splash.

Docking layouts, shipped presets, and the few keyboard gestures that exist in the shell.

Layout files

Location	Role
`C:\Program Files\Quanto\Layouts\Default.qtl`	Factory workspace preset
`%LOCALAPPDATA%\Quanto\Layouts\...`	Live docking, presets, autosaves for MainWindow and MethodEditor

A `.qtl` is a zip of docking layout plus Fast grid JSON. Drop a `.qtl` onto Quanto to apply it. Layouts do not contain results. Folder map: Installation and folders.

Dockable panels

The main shell uses dockable panels (samples, results, chromatograms, reports, method tables, and so on). Save a comfortable arrangement as a user preset under Local AppData; reset toward `Default.qtl` when needed.

Grid column fill, sort, freeze, and navigator: Grids.

Keyboard / shortcuts

A search of the WPF host did **not** find a general `KeyBinding` shortcut map for the main shell. Day-to-day UI is **menu / ribbon / toolbar** driven.

Known gestures:

Gesture	Where	Effect
Shift during splash	Startup	Recycle settings, layouts, and/or licenses (Recycle Bin)
Enter	Find / dialogs	Accept / find (dialog Enter)
Ctrl+wheel	FastSettings	UI zoom
Report Designer	Designer window	Ctrl+B / I / U / X / C; Ctrl+N / O / S; Ctrl+F3 (designer editing, not the main shell)

Do not assume undocumented accelerator keys for Analyze, Save, or Key Review.

Related

- Getting started
- Grids
- Installation and folders

QUANTO USER GUIDE

Troubleshooting

MSTree2/TDA, MH vs CS, SCIEX MRM, Auto mode, add-ins, license save.

Common extract, add-in, and license problems. For install layout see Installation and folders. For vendor formats see ChemStation and SCIEX.

MassHunter: AcqData, MSTree2.bin, TDA converter

MassHunter `.D` samples need an **AcqData** folder. Indexed extract looks for:

`sample.d\AcqData\MSTree2.bin`

If that file is missing, Quanto looks for Agilent:

`C:\Program Files\Agilent\MassHunter\Workstation\Quant\bin\TDAConverterConsole.exe`

and writes the index **into the sample** `.D`. The Windows account needs **write** permission on the `.D` folder. Prefer enabling indexed-data export on the instrument so conversion already happened.

TDA conversion is MassHunter only. ChemStation `.D` does not use TDA — it reads natively via MSDChem (ChemStation vendor notes).

ConvertResult outcomes (MassHunter index)

When Quanto runs or skips conversion, typical outcomes are:

Outcome	Meaning
Success	Index written; extract can proceed
Already indexed	`MSTree2.bin` (or equivalent) was already present
Empty / converter missing	`TDAConverterConsole.exe` not found — install MassHunter Quant or pre-index elsewhere
Error strings	Corrupt `MSScan` / TDA failure messages — repair or re-acquire the `.D`, or re-run conversion on a writable copy

Write permissions on `.D`

Symptom	Check
Convert fails / index not written	NTFS write on the sample `.D` (and parent share)
Read-only network snapshot	Copy to a writable local or study folder, or pre-index on the acquisition PC

ChemStation vs MassHunter `.D` provider

Both use `.D` folders, but different add-ins and file layouts:

Kind	Provider	Needs	Conversion
MassHunter	`AgilentMassHunterDataProvider`	`AcqData\` (+ `MSTree2.bin` for indexed access)	May need TDAConverterConsole
ChemStation / MSD	`AgilentChemStationDataProvider`	`data.ms` and/or `dataSim.ms`	Native MSDChem — **no** TDA

Wrong expectations (waiting for TDA on ChemStation data, or opening MH data without AcqData/index) cause empty or failed extracts. Confirm which vendor created the folder.

SCIEX: MRM-centric extract

SCIEX WIFF extract is **MRM-centric** (transitions from the WIFF via Clearcore2). Empty or unexpected chromatograms often mean the method ions do not match what is in the file, or the multi-sample `#n` selection is wrong. See SCIEX vendor notes.

Empty chromatogram in Auto mode

Internal acquisition mode is `Auto`, `Scan`, `Sim`, or `MsMs`. **Auto** detects from the file and tries alternates if a channel comes back empty. If the chrom stays empty:

1. Confirm the Quant ion / transition exists in that sample.
2. Force Scan / Sim / MsMs on the Quant if Auto picked the wrong mode.
3. Confirm the correct data provider (MH vs ChemStation vs SCIEX).
4. For MH, confirm `AcqData\MSTree2.bin` exists or conversion succeeded (`Success` / `Already indexed`).
5. For ChemStation, confirm `data.ms` / `dataSim.ms` are present.

Add-in failed to load

Add-ins load from `C:\Program Files\Quanto\AddIns\`:

Folder	Role
`DataProvider\`	Raw data readers
`Integrator\`	Peak detection
`MethodImporter\`	Vendor methods
`VendorSupportFiles\Agilent\` / `Sciex\`	Native vendor DLLs

Checks:

- Whole `AddIns` tree matches this Quanto build (do not mix versions).
- In-app logbook / crash Recovery zip under `%LOCALAPPDATA%\Quanto\Recovery\` after a failed start.
- Dependent runtimes (.NET 10 Desktop x64; vendor support files present).

See also Creating add-ins.

License blocks save

After the 21-day installation grace, save is blocked when recent samples (acquired within 30 days) lack an instrument serial license. Unlicensed serials are highlighted earlier. Details: Licensing.

Cloudflare / web cache

Quanto is a **desktop** app. Browser Cloudflare cache, CDN, or site cookies do **not** affect Analyze, licenses, or local `.qtb` files. Website downloads of example zips are separate from the running application.

QUANTO USER GUIDE

Creating add-ins

Data provider, method importer, and integrator contracts; load folders and bootstrap.

Quanto loads **three kinds of add-ins** at startup from folders next to the app. Drop a DLL in the right folder; no registry keys.

Kind	Folder	Contract	Job
Data provider	`AddIns\DataProvider\`	`IDataProvider`	Find samples, read bottle metadata, extract chromatograms
Method importer	`AddIns\MethodImporter\`	`IMethodImporter`	Find and parse vendor methods into compounds / signals / levels
Integrator	`AddIns\Integrator\`	`IIntegrator`	Detect peaks on `x[]` / `y[]`

Contracts live in `Quanto.Shared.Core` (`Quanto.Shared.Core.Interfaces`). The host loader is `Quanto.AddInFactory.AddInLoader`.

How loading works

1. At startup the host calls `AddInLoader.Load(integratorPath, dataProviderPath, methodImporterPath)`.
2. Each folder is scanned for `*.dll`. Assemblies are `Assembly.LoadFrom`.
3. Instantiation is **lazy** — constructors (and heavy vendor init) run on first use, not at discovery.
4. Registration key is the **concrete class name** (e.g. `AgilentMassHunterDataProvider`, `Quanto`, `SciexWiffMethod`).
5. Optional `IAddIn`: `AlternativeDllPath` (vendor DLL search folder) and `Version` (for logs).
6. Optional **bootstrap** class: a tiny type with an `AlternativeDllPath` property (conventionally named `AddInBootstrap`, or any `*Bootstrap`). Instantiated first so vendor paths are registered **before** reflecting types that reference vendor assemblies.

Paths (from `AppPaths`):

```
text {app}\AddIns\ DataProvider\ MethodImporter\ Integrator\ VendorSupportFiles\Agilent\ VendorSupportFiles\Sciex\
```

Relative `AlternativeDllPath` values are resolved from the application directory (examples from shipped add-ins: `AddIns\VendorSupportFiles\Agilent`, `AddIns\VendorSupportFiles\Sciex`).

Shipped projects copy only their **own** primary DLL into the staging add-in folder; vendor natives stay under `VendorSupportFiles\…`.

Project setup

- Target **Windows x64** and the same Quanto Windows TFM as the host (.NET 10 Windows).
- Reference `Quanto.Shared.Core` only for contracts and shared types/enums (do not take a hard dependency on `Quanto.WPF`).
- Implement the matching interface(s) on a **public non-abstract class**.

- Optionally implement `IAddIn`.
- Add a dependency-free bootstrap type if you need vendor DLLs on the probe path before your main type loads.
- Build output: one add-in DLL into the correct `AddIns\…` subfolder.

Example bootstrap (duck-typed; implementing `IAddInBootstrap` is fine but not required for discovery):

```csharp namespace MyVendorData;

public sealed class AddInBootstrap { public string? AlternativeDllPath => @"AddIns\VendorSupportFiles\MyVendor"; } ```

## Data provider

`IDataProvider` extends:

- `ISampleFinder` — `HashSet<SampleFile> FindSamples(string path)`
- `ISampleFileReader` — `void ReadSampleFile(SampleFile sampleFile)`
- `ISignalReader` — chromatogram extraction

```csharp public interface ISignalReader { SignalPoints GetSignalPoints( string filePath, double rt, double beginOffsetTime, double endOffsetTime, double mass1, double beginOffsetMass1, double endOffsetMass1, double mass2, double beginOffsetMass2, double endOffsetMass2, IonPolarity? ionPolarity = null, double? fragmentorVoltage = null, double? collisionEnergy = null, ChromatogramAcquisitionMode preferredMode = ChromatogramAcquisitionMode.Auto);

`SignalPoints GetTotalSignal(string filePath);`
`// optional override:`
`SignalPoints GetTotalSignal(string filePath, ChromatogramAcquisitionMode mode);`

} ```

`ChromatogramAcquisitionMode`: `Auto`, `Scan`, `Sim`, `MsMs`.

`FindSamples` should return `SampleFile` instances with `DataProvider` set to this instance and fill what you can (name, AcqDateTime, instrument, position, dilution, dose hints). The host matches sample type / dose from the method afterward.

**Optional host hooks** (implement if useful): `IMrmTransitionCatalog`, `IDataFilePrefetch`, `IDataFileRelease`.

### Reference implementations

| Class | Project | Format |
|---|---|---|
| `AgilentMassHunterDataProvider` | `Quanto.AddIn.DataProvider.AgilentMHData` | MassHunter `.D` |
| `AgilentChemStationDataProvider` | `Quanto.AddIn.DataProvider.AgilentCSData` | ChemStation `.D` (native MSDChem) |
| `SciexWiff` | `Quanto.AddIn.DataProvider.SciexWiff` | SCIEX `.wiff` |

## Method importer

`csharp public interface IMethodImporter { string Name { get; set; } HashSet<MethodFile> FindMethods(string path); void LoadMethod(string filePath); void ReadMethodFile(MethodFile info); ICompound[] GetCompounds(); }`

Map vendor content onto `ICompound` / `ISignal` / `ILevel`:

- Compound: name, group, expected time, begin/end offsets, ISTD flag and link, curve fit/weight/origin, concentration limits, signals, levels.
- Signal: name, `Mass1` / `Mass2`, collision energy, fragmentor, ion polarity, `AcquisitionMode`, quant/qualifier flags, % and tolerance.
- Optional `ICompoundWithAdditionalQuants` when the vendor has extra quant slices (e.g. MassHunter RA variants).

`FindMethods` should attach `MethodImporter = this` on each `MethodFile` and set a display `Name`.

### Reference implementations

| Class | Project | Source |
|---|---|---|
| `AgilentMhMethod` | `Quanto.AddIn.MethodImporter.AgilentMHMethod` | `.m` / `.quantmethod.xml` |
| `AgilentMassHunterDataMethod` | `Quanto.AddIn.MethodImporter.AgilentMHData` | Method from `.D` |
| `AgilentCsMethod` | `Quanto.AddIn.MethodImporter.AgilentCSMethod` | `.M` + `qdb.mth` |
| `SciexWiffMethod` | `Quanto.AddIn.MethodImporter.SciexWiff` | Method from `.wiff` |

## Integrator

`csharp public interface IIntegrator { PeakValues[] Integrate(double[] x, double[] y, string settings); }`

`PeakValues` fields: `BegX`, `EndX`, `BegB`, `EndB` (peak time bounds and baseline height at those bounds).

`settings` is the free-form string from the channel's integrator settings. Parse what you need; ignore unknown keys.

Channels pick an integrator **by class name** (default shipped name: `Quanto`).

### What the integrator returns vs what Quanto calculates

An integrator **only detects peak boundaries**. It does **not** compute area, height, apex time, width, or symmetry.

After your add-in returns `PeakValues[]`, the host always runs **Quanto's shared peak-metric methods** in `Quanto.Shared.Core.PeakCalculations` (`IPeakMetricsCalculator` / `PeakMetricsCalculator`) on the same `x[]` / `y[]` and your `BegX` / `EndX` / `BegB` / `EndB`. That path is used for:

- automatic integration (`ImportPeakValuesService` after any integrator, including custom ones)
- manual re-integration and boundary edits (`PeakIntegrationService`)

Derived metrics include (see `PeakMetrics`):

| Metric | Meaning |
|---|---|
| `Area` | Trapezoidal integral of the **baseline-corrected** signal |
| `Height` | Apex height above the linear baseline |
| `CenterX` / `Time` | Apex retention time |
| `MaxSignal` | Apex signal value |
| `FwAt50`, `FwAt5` | Peak widths at 50% and 5% of height |
| `Symmetry` | Peak symmetry from those crossings |
| `BegR`, `EndR` | Residuals / edge refinements from the shared calculator |

Response for quantitation is then **area** or **height** according to the analyte's `QuantitateByHeight` flag — still from these shared metrics, not from the integrator DLL.

So when you write an integrator:

1. Focus on finding good start/end times and baseline points.
2. Return accurate `BegX`, `EndX`, `BegB`, `EndB`.
3. **Do not** reimplement area/height in the add-in; Quanto will calculate them consistently for every integrator (Quanto, Slice, or yours).
4. You may call `PeakMetricsCalculator.Instance` yourself for diagnostics or unit tests (same algorithms), but the host will recalculate when importing peaks into the batch.

### Reference implementations

| Class | Project | Notes |
|---|---|---|
| `Quanto` | `Quanto.AddIn.Integrator.Quanto` | Production peak detector (boundaries only; metrics still via shared calculator) |
| `Slice` | `Quanto.AddIn.Integrator.Slice` | Minimal stub (returns no peaks) — useful as a template |

## Minimal skeletons

### Integrator

```csharp using Quanto.Shared.Core.Interfaces; using Quanto.Shared.Core.Types; // Optional for offline checks: // using Quanto.Shared.Core.PeakCalculations;

namespace MyIntegrator;

public sealed class MyIntegrator : IIntegrator, IAddIn { public string? AlternativeDllPath => null; public string? Version => "1.0.0";

```
public PeakValues[] Integrate(double[] x, double[] y, string settings)
{
    // Detect peaks; return only BegX/EndX/BegB/EndB.
```
```

```
    // Host applies PeakMetricsCalculator for Area, Height, Fw*, Symmetry, etc.
    return [];
}

} ```
```

Data provider / method importer

Follow the same pattern: implement the interface + optional `IAddIn`, discover files under the path the host passes, and return shared types from `Quanto.Shared.Core.Types` / `Enums`. Prefer looking at the matching Agilent or SCIEX project above rather than re-deriving vendor binary formats.

Checklist before shipping an add-in

- [ ] DLL builds x64 / Quanto Windows TFM and copies to the correct `AddIns\…` folder
- [ ] Class is public, non-abstract, and assignable to the right interface
- [ ] Class **name** is unique among loaded add-ins of that kind (registry key)
- [ ] Bootstrap + `AlternativeDllPath` set if vendor natives are required
- [ ] Vendor DLLs installed under `VendorSupportFiles\…` (not duplicated beside every add-in)
- [ ] Constructors avoid crashing the process at first use (lazy load still runs your ctor once)
- [ ] Version visible via `IAddIn.Version` or assembly informational version
- [ ] Smoke-test: host log shows Discovered / instantiated lines; Find / Load / Integrate works on a real file
- [ ] Integrators: confirm Area/Height appear after Analyze (shared `PeakMetricsCalculator`), not only raw boundaries

Related

- Formats and modes: Compatibility
- Install tree: Installation and folders
- Analyst import UI: Reports, import, and export