

View add-in SDK

Custom review panes under View → Add-in views; IViewAddIn, Review, and hot-drop.

A **view add-in** is a .NET DLL in `AddIns\View`. Quanto finds every public `IViewAddIn`, shows it under **View → Add-in views**, and docks the pane in the same shell as built-in views. The add-in describes the view. The host owns the dock, the theme, and the batch.

Extend Quanto with custom review panes without leaving the app. Build once, drop on the window, keep reviewing.

Example view add-ins ship with Quanto under `AddIns\View` (for example Showcase, Sample review, FastPositive, Key Review, and Nav selection). Source samples such as Showcase and SampleReview are available to partners on request.

What you reference

Reference	When
<code>Quanto.AddIn.Sdk</code>	Always. This is the public contract. It references nothing else in Quanto.
<code>Fast(Fast.Grid , Fast.Wpf , Fast.Flow , Fast.Core)</code>	When the view is a screen that creates Fast controls.

Do not reference `Quanto.WPF`, `Quanto.WPF.Views`, `Quanto.Core`, `Quanto.Shared.Core`, `Quanto.GridViewModels`, or `Quanto.Services`. Those are the application. A view you ship does not need them, and a build of Quanto does not need your DLL.

The surface

Five names.

`IViewAddIn` — `Id` and `Describe()`. `Id` is the dock key. Make it an expression-bodied property so the loader can read it without running the constructor. No `|`, dot, slash, or surrounding space. Do not reuse a built-in view class name.

`View.Named(id, title)` — builds a `ViewSpec`.

- `Grid` — one or more Fast grids. The host creates the grid.
- `Surface` — a named pane. Pass a grid builder when the pane also has columns.
- `Screen` — your code creates the content. The host docks it.
- `Panel` — side panel. The host paints Fast controls from the descriptors and restores checks and choices.
- `Many` — the View menu may open another pane. The default is one.
- `Remember` — one extra string (last action, selected mode), saved beside the panel values.

A view is either panes or a screen, not both.

`Review` — the open batch as rows, filters, and commands.

- `Selection` — relative sample paths, analyte names, parallel acquisition keys, quant keys, result ids, and the focused sample, analyte, and result. `Batch.CurrentKey` is the sample key the host follows.
- `Samples`, `Results`, `MatchingSamples`, `MatchingResults`
- `Checked`, `Choice`
- `Approve`, `Reject`, `Clear`, `AcceptOnly`, `ClearConcentration` — one list of result ids. The host applies the whole list inside one update, recomputes sample-key selection once, and notifies once. These stay the pin/reject API; do not encode that decision as `ApplyProperties`.
- `ApplyProperties` — one batch of sample, compound, and result column writes (`PropertyWrite`). The host notifies once at the end. A read-only batch (package, or opened without the save lease) writes nothing. Showcase includes a stamp-comments example.
- `Batch` — samples, compounds, quants, analytes, tiles, formats
- `Cover`, `RejectGroup`, `GroupRejected`
- `Refresh`

Result ids belong to that snapshot. The next refresh uses new ids.

`IViewScreen` / `IScreenSite` — only when the grid shell is not enough.

- `Element` is the control the host docks (`FrameworkElement` at runtime). The SDK itself does not reference WPF.
- `Attach` binds a Fast control to the host theme, language, and zoom.
- `Bind` loads and saves the column layout of a `Fast.Grid.GridControl` (widths, order, visibility).
- `OnReview` runs on the UI thread after the host has restored `State` and `Keep` values.
- `Publish` tells the rest of Quanto which samples, analytes, quant keys, and result ids are selected.
- `Segments("supercell")` is the enabled SuperCell field list.
- `Text` is the Settings language string. The second argument is the English fallback.
- `State` is the extra string (selected tab, last action).
- `Setting` / `Keep` store a screen's own control values (which boxes are checked, which dropdown is selected). The host restores them before `OnReview`.

Layout — file names. You rarely call it. The Notes pane in Showcase prints them.

- Content id: {id}|{title}
- Column layout, one grid: {title}-FastLayout-{id}.json
- Column layout, pane or screen grid: {title}-FastLayout-{id}.{paneId}.json
- State: {title}-ViewAddInState-{id}.json

The host restores column widths, order, and visibility from the Fast layout files, the same way a built-in grid does. It restores panel checkboxes and dropdowns, the selected tab, the Remember string, and Keep values from the state file before the first row fill. A missing control entry uses the default from Describe (or, for a screen, the value the screen uses when Setting returns null). A check is stored as true or false . A choice is stored as the choice id. Remember stays the extra string. It is not where filters go.

In the SDK

IViewAddIn , View , ViewSpec , ViewPane , ViewColumn , ViewRow , PanelBuilder , ViewControl , ViewState , Review and its rows (SampleRow , ResultRow , BatchGraph , QuantNode , TileNode , Pick , PropertyWrite), IViewScreen , IScreenSite , IGridBinding , GridColumn , CellSegment , Layout .

Host only

Dock panes, Fast grid hosts, theme bootstrap, the batch and quantitation model, sample-key selection, the isomer-sum algorithm behind Review.Cover , and the projection that fills Review . Add-in code does not see those types.

Integrator, data-provider, and method-importer contracts stay on Quanto.Shared.Core . A view add-in does not need that assembly. Those other add-in kinds are a different install path — they are not loaded by the view hot-drop.

Examples that ship

	FastPositive	Key Review	Sample review	Showcase	Nav
Id	FastPositiveAddIn	KeyReviewAddIn	SampleReviewView	ViewShowcase	Nav
Title	FastPositive	Key Review	Sample review	Showcase	Nav
DLL	AddIns\View\FastPositiveAddIn.dll	AddIns\View\KeyReviewAddIn.dll	AddIns\View\SampleReview.dll	under AddIns\View\	und Add
Content	Concentration tiles for every sample that passes the sidebar and navigator filters, on every sample key.	Matrix for the current sample key, Accept / Reject / Exclusive Accept / Reset / Clear ManFConc, Group Sum side panel.	One-grid review with Approve / Reject and panel filters.	Composed shell: Samples, Results, Notes, empty Screen surface, shell panel, Results filter.	One diag the i sele

FastPositive and Key Review implement IViewScreen . They paint from Review.Batch . Decisions go through Review . Group Sum calls Cover and RejectGroup . Open Key Review to use the side panel.

Prefer Showcase and SampleReview when you learn by example. They show grids, panels, Approve / Reject / Clear, and property writes through the host.

Ship

Change the assembly name and Id from an example. Keep Id stable after you ship. Build x64. Place the add-in DLL in AddIns\View\ next to the app (no registry).

- Restart Quanto after copying, or
- Hot-drop — drop the DLL, a build output folder, or a zip onto the running main window. Quanto copies into AddIns\View , replaces a loaded view with the same IViewAddIn.Id , and opens or rebuilds its dock. Layout files stay keyed by id. The status bar confirms success. A drop that is not a view add-in is rejected with a dialog; existing views stay loaded.

Quanto.AddIn.Sdk.dll is not part of the View drop. It lives with the host next to the Quanto executable. At run time the SDK and Fast assemblies come from the host. You do not need the Quanto application source to compile — you need Quanto.AddIn.Sdk and, for a screen, the Fast assemblies you compile against.

There is no license check on this surface. Selling the DLL is a packaging and contract question, not an SDK callback.

See also Creating add-ins.

Teaching sample

Showcase references only the SDK. It is the composed shell: Samples, Results, Notes, an empty Screen surface, a shell panel, and a Results filter. Approve, Reject, and Clear call Review with the visible result ids. Notes prints the file names from Layout . Stamp-comments shows ApplyProperties / PropertyWrite .

SampleReview is the one-grid example.

Nav selection is a one-grid diagnostic of the navigation selection. It refreshes with the review and shows relative sample paths, acquisition key + analyte name, quant keys, result ids, the focused sample, analyte, and result, `Batch.CurrentKey` , and each selected sample row.

Installed examples land as `AddIns\View\*.dll` (for example `SampleReview.dll` / `FastPositiveAddIn.dll`). Source for Showcase and `SampleReview` is available to partners on request.