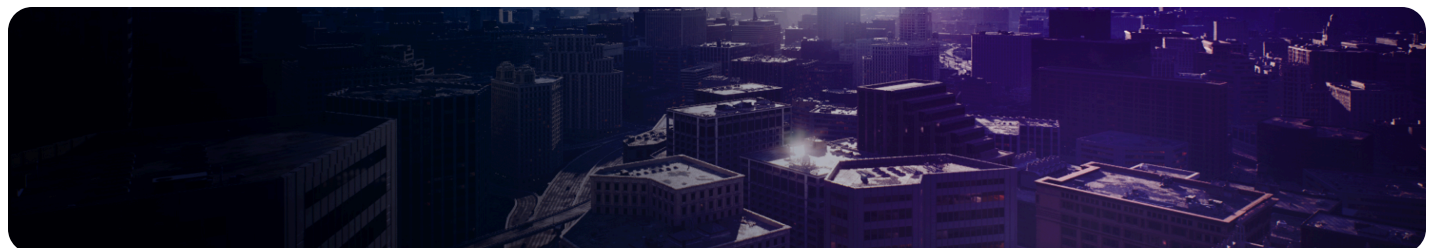


Invalidation Box

An overview of Invalidation Box, which helps developers optimize UI Widgets.



PREREQUISITE TOPICS



In order to understand and use the content on this page, make sure you are familiar with the following topics:

- [Widget Type Reference](#)

Description

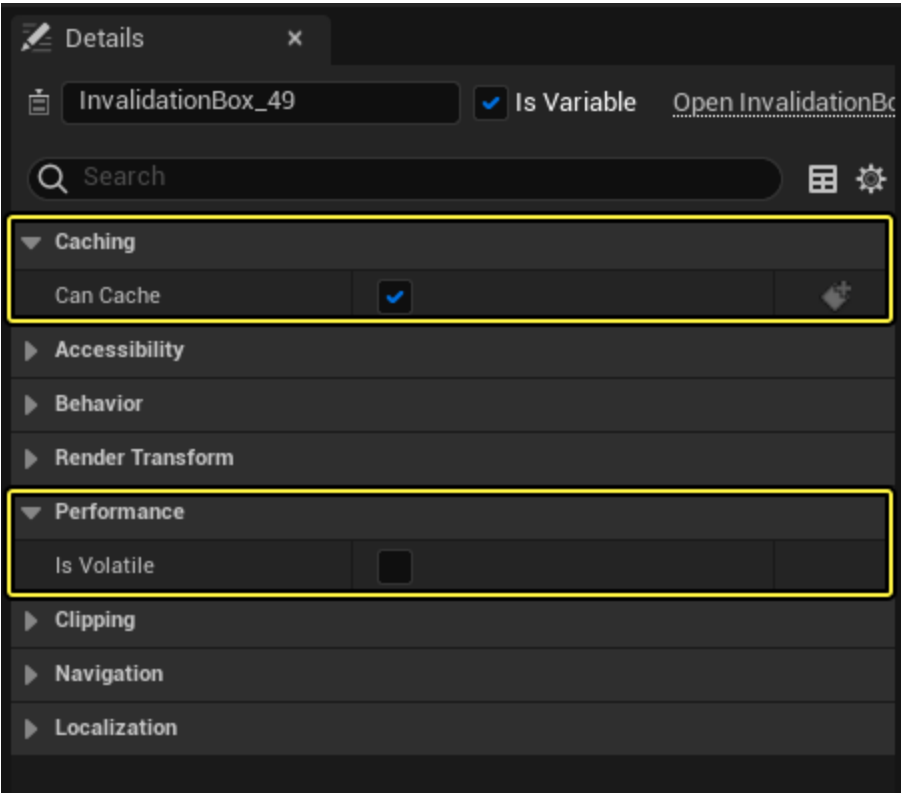
Widgets that are wrapped with an **Invalidation Box** allows the child widget geometry to be cached to speed up Slate rendering. Any widgets that are cached by an Invalidation Box are not pre-passed, ticked, or painted. In general, if you are looking to optimize your project, wrapping certain widgets with Invalidation Boxes may boost your performance (particularly for mobile projects or complicated UI displays). For widgets that do not change constantly, they can be placed inside an Invalidation Box and cached instead of considered during paint, tick, or prepass.

If the widget changes, however, it will become invalid and you will need to manually invalidate the cache which will throw it away essentially and force it to redraw on the next paint pass.

Anything that changes the visual appearance of the widget requires it to be invalidated. The only exception to this is a change to the appearance that is not stored in the vertex buffer for those widgets (for example Materials, as changing a Material Parameter does not require invalidating the widget).

Details

In the **Details** panel for a placed **Invalidation Box**, there are a couple of specific options that can be set that pertain to the widget:



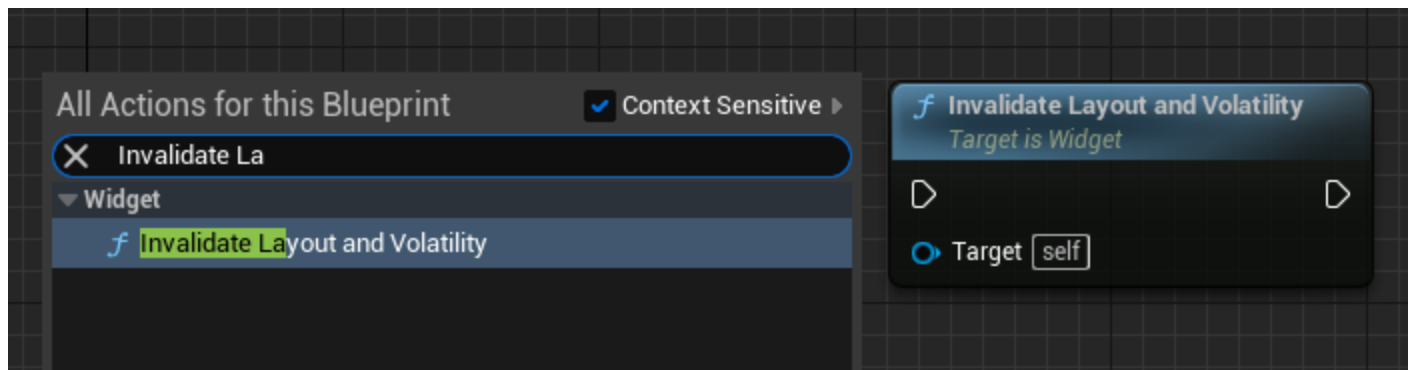
Option	Description
Cache Relative Transforms	Caches the locations for child draw elements relative to the invalidation box which adds extra overhead to drawing them every frame. However, in cases where the position of the invalidation boxes changes every frame, this can be a big saving.
Is Volatile	If true, prevents the widget or its child's geometry or layout information from being cached. If this widget changes every frame, but you want it to still be in an invalidation panel, you should make it as volatile instead of invalidating it every frame,

Option	Description
	which would prevent the invalidation panel from actually ever caching anything.

Regarding the **Is Volatile** check box, any widget can be set to be volatile. Volatile widgets act like normal Slate widgets pre-invalidation. They're redrawn every frame, including all their children. When combined with the invalidation panel, it allows you to care only about redrawing the most dynamic bits of the UI, as invalidating a whole panel could be far more costly.

Functions

When using an **Invalidation Box**, it is up to the user to call `Invalidate` through C++ or the **Invalidate Layout And Volatility** node (pictured below) on a child widget to force invalidation.



Currently, some core widgets do this automatically when certain properties are changed, however more will do it over time.

Debugging

You can debug your Invalidation Boxes using the **Widget Reflector** (CTRL+Shift+W) and clicking the **Invalidation Debugging** toggle.



To display the legend when `SlateDebugger.Invalidate.Enable 1` is invoked, use `SlateDebugger.Invalidate.ToggleLegend`.

Widget Name	FG	Visibility	Widget Info	Address
SWindow		Visible	PlayLevel.cpp(2980)	0x000000009D5E3900
SOverlay		SelfHitTestInvisible	SWindow.cpp(439)	0x00000000C9F4D800
SImage		SelfHitTestInvisible	SlateApplication.cpp(3747)	0x00000000C3922500
SImage		SelfHitTestInvisible	SlateApplication.cpp(3747)	0x00000000C3922380
SVerticalBox		SelfHitTestInvisible	SWindow.cpp(465)	0x00000000C9F4DD00
SVerticalBox		SelfHitTestInvisible	SWindow.cpp(386)	0x000000003004C900
SWindowTitleBar		SelfHitTestInvisible	SlateApplication.cpp(3756)	0x00000000ACD4C100
SViewport		Visible	PlayLevel.cpp(3017)	0x000000008F613380
SGameLayerManager		SelfHitTestInvisible	PlayLevel.cpp(3010)	0x00000000ACD4CA00
SScissorRectBox		SelfHitTestInvisible	SGameLayerManager.cpp(55)	0x00000000C9F4F4C0
SDPIScaler		SelfHitTestInvisible	SGameLayerManager.cpp(27)	0x00000000C3920100
SOverlay		SelfHitTestInvisible	SGameLayerManager.cpp(30)	0x00000000C9F4FB00
SCanvas		SelfHitTestInvisible	SGameLayerManager.cpp(34)	0x00000000C9F4EAC0
SOverlay		SelfHitTestInvisible	PlayLevel.cpp(3008)	0x00000000C9F4FC40
SConstraintCanvas		SelfHitTestInvisible	UserWidget.cpp(425)	0x000000009455B280
SObjectWidget		SelfHitTestInvisible	ExampleWB [ExampleWB_C_57]	0x00000000944A6E00
SConstraintCanvas		SelfHitTestInvisible	ExampleWB [CanvasPanel_1]	0x000000009455BB80
SInvalidationPanel		SelfHitTestInvisible	ExampleWB [InvalidationBox_45]	0x00000000ACD4E200
SBorder		Visible	ExampleWB [Border_0]	0x00000000B9CC7880
SPopup		SelfHitTestInvisible	SGameLayerManager.cpp(44)	0x00000000A6696080
SPopupLayer		SelfHitTestInvisible	SWindow.cpp(479)	0x00000000C9F4D440
SImage		SelfHitTestInvisible	SlateApplication.cpp(3747)	0x00000000C3922680
SHorizontalBox		SelfHitTestInvisible	STutorialRoot.cpp(56)	0x0000000078CFEC0

[Click image for full view.](#)

With the Widget Reflector up and Invalidation Debugging on, you will see the following colors:

Color	Description
Yellow	Paint invalidated this frame.
Gray	Volatility invalidated this frame.
Cyan	ChildOrder invalidated this frame.
Black	RenderTransform invalidated this frame.
White	Visibility invalidated this frame.
Pink	Layout invalidated this frame for Invalidation Box.
Blue	Every widget was reordered.

Color	Description
Red	Fully re-updated.



To debug InvalidationBox behavior, use `SlateDebugger.InvalidationRoot.Enable`.

Example below shows an image that is marked as volatile inside of a Border which is wrapped with an Invalidation Box. Since the image is marked as volatile, it can be updated dynamically during gameplay while the Border (or any other art assets that you wanted to appear around the image that do not change) is cached.

