Predefined Script Tasks — Client Lifecycle Matrix (10.1.5 draft)

<ac:layout><ac:layout-section ac:type="single"><ac:layout-cell> Per-flavor execution contract for the four predefined script tasks. <ac:link><ri:page ri:content-title="Technical Reference" /><ac:plain-text-link-body><![CDATA[Reference]]></ac:plain-text-link-body></ac:link> → <ac:link><ri:page ri:content-title="Platform Modules Reference" /><ac:plain-text-link-body><![CDATA[Modules]]></ac:plain-text-link-body></ac:link> → <ac:link><ri:page ri:content-title="Scripts Module Reference" /><ac:plain-text-link-body><![CDATA[Scripts]]></ac:plain-text-link-body></ac:link> → <ac:link><ri:page ri:content-title="Scripts Tasks Reference" /><ac:plain-text-link-body><![CDATA[Tasks]]></ac:plain-text-link-body></ac:link> → Client Lifecycle Matrix <hr /><ac:structured-macro ac:name="info" ac:schema-version="1"><ac:rich-text-body> 10.1.5 DRAFT. Preview content for the FrameworX 10.1.5 release. The parent page will be updated to incorporate this matrix at release time. See TDEV-1325. </ac:rich-text-body></ac:structured-macro><h2>Lifecycle Matrix</h2> FrameworX has four predefined script tasks that fire on platform lifecycle events. Each task has a fixed execution surface depending on which client flavor is connecting: <table class="wrapped"><tbody><tr><th> Predefined Task </th><th> WPF rich client </th><th> HTML5 / web client </th><th> Notes </th></tr><tr><td> ServerStartup </td><td> n/a </td><td> n/a </td><td> Runs on the server (TServer.exe) at solution start, before any client connects. </td></tr><tr><td> ServerShutdown </td><td> n/a </td><td> n/a </td><td> Runs on the server at solution stop. Reliable — server shutdown is graceful. </td></tr><tr><td> ClientStartup </td><td> Yes </td><td> Yes </td><td> Runs once per client at session connect, on the client process. New in 10.1.5: the HTML5 / WebAssembly client now executes this task too. WPF behavior unchanged. </td></tr><tr><td> ClientShutdown </td><td> Yes </td><td> No (by design) </td><td> Runs at client close. Browser tab close gives the HTML5 client no reliable shutdown window — tab kill, browser kill, navigation away, OS eviction on mobile, and device sleep all skip the unload phase. Wiring an unreliable hook would lie to script authors about determinism, so the platform deliberately does not invoke ClientShutdown on the web client. </td></tr></tbody></table><h2>Designing for Web Compatibility</h2><h3>If you need teardown logic to fire reliably on web</h3> Use the ServerShutdown task or a Server-domain script class instead. The server detects client disconnects and can perform any must-finish work centrally — saving state, releasing resources, persisting a session record. <h3>If a ClientShutdown task body is non-empty in a web-targeted solution</h3> The Designer's Build & Publish dialog emits a one-line info note in the build log (10.1.5 and later) so authors who copy a WPF solution to web don't silently lose teardown logic. Read it before you ship the web build. <h3>API surface for ClientStartup on web</h3> Client-domain task code is compiled twice — once for the rich-client target (.NET Framework 4.8) and once for the web target (WebAssembly-safe surface). WPF-only APIs (<code>System.Windows.MessageBox.Show</code>, <code>System.Diagnostics.Process.Start</code>, file-system access, registry, etc.) fail at publish time, not at runtime. A clean web publish is the contract that the script will run. <h2>Background</h2> The decision to make ClientShutdown rich-client-only on the web reflects a deliberate platform stance: contracts that can fire unreliably are worse than contracts that don't fire at all. A "best-effort" ClientShutdown would silently skip in the most common shutdown paths a real user creates (closing the browser, switching tabs aggressively, OS evicting a backgrounded tab on mobile), and any code written under the assumption that it runs would be subtly incorrect. By contrast, the WPF rich client closes through a graceful <code>MainWindow.Closing</code> path that the platform can hook synchronously, so ClientShutdown fires deterministically there. <hr /><h4>In this section...</h4> <ac:structured-macro ac:name="pagetree" ac:schema-version="1"><ac:parameter ac:name="root"><ac:link><ri:page ri:content-title="@parent" /></ac:link></ac:parameter></ac:structured-macro> <hr /> </ac:layout-cell></ac:layout-section></ac:layout>