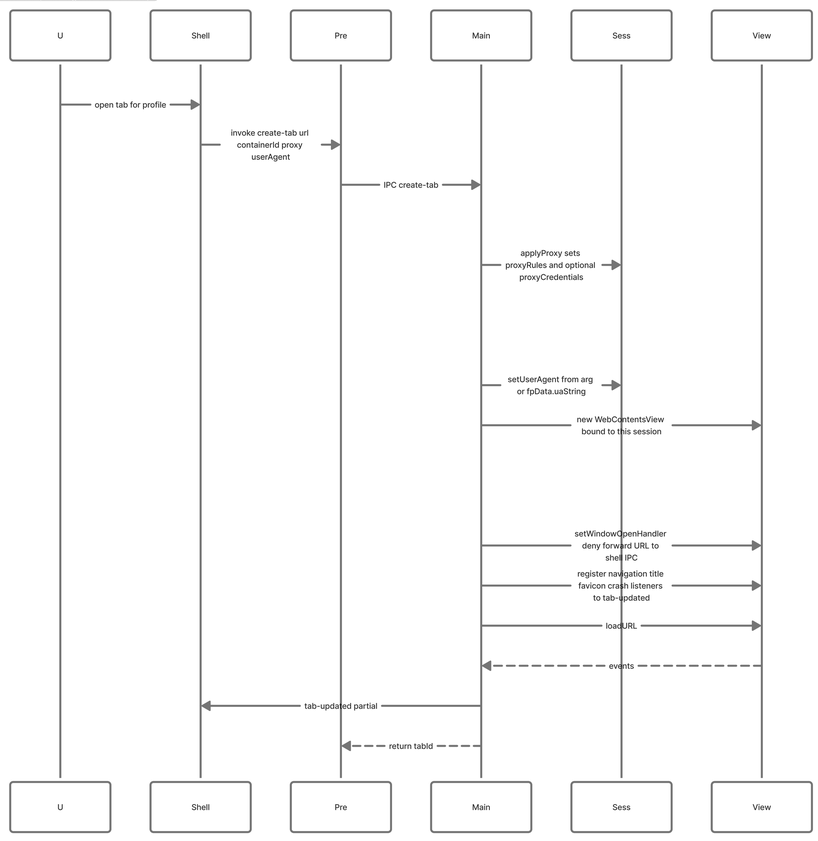
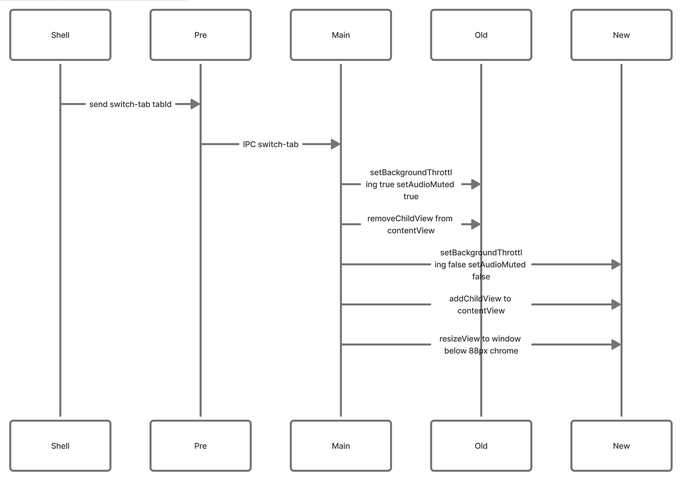
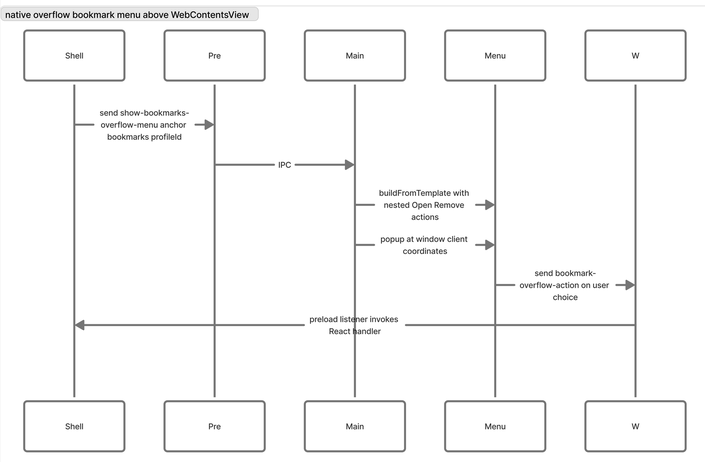
create-tab (ordering and side effects)



switch-tab (single visible WebContentsView)



native overflow bookmark menu above WebContentsView



Why some UI is native Electron Menu

Bookmark overflow list uses Menu.popup so it appears above WebContentsView without resizing the view

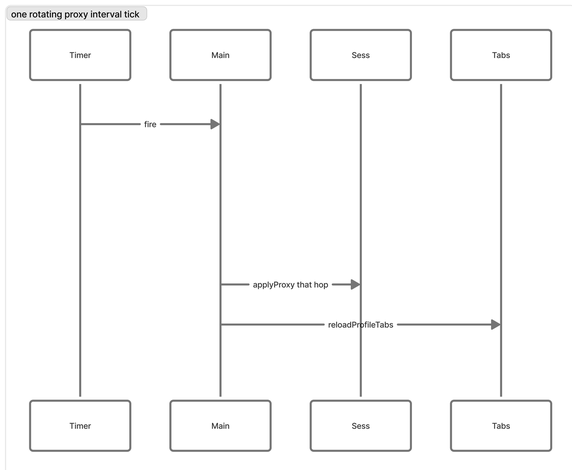
Shell renderer — authoritative UI state

- activeTabId: must stay in sync with switch-tab IPC
- profiles: React state mirrored from localStorage on load; every change triggers saveProfiles effect
- tabs: TabInfo array — ids from create-tab; url title favicon loading crashed merged from tab-updated events
- profilesRef tabsRef activeTabIdRef addTabRef — avoid stale closures in long-lived IPC listeners

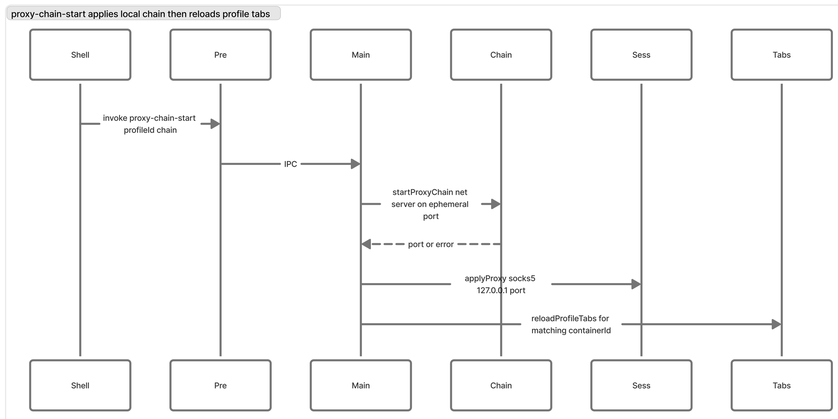
Hard invariants & platform constraints

- Chrome height contract: 30 + 30 + 28 = 88 px — must match resizeView y offset and App chrome stack
- Single BrowserWindow: shell webContents loads React UI; tab pages live in WebContentsView children of contentView
- webRequest onBeforeSendHeaders hook is registered at most once per partition via acceptLanguageHookPartitions Set
- WebContentsView is composited above shell DOM: HTML/CSS cannot paint over the page area without native menus or moving the view
- Each profile id maps to one persisted Electron session partition string persist:containerId — cookies/cache isolated per profile
- Fingerprint vector is cached per containerId in main process fingerprintCache — stable across tabs of same profile until last tab destroyed

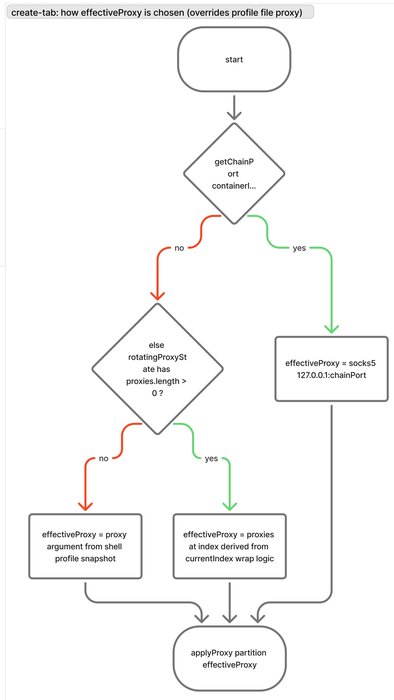
one rotating proxy interval tick



proxy-chain-start applies local chain then reloads profile tabs



create-tab: how effectiveProxy is chosen (overrides profile file proxy)



create-tab: ordered side effects before first byte matters

1. tabId = uuid partition = persist containerId
2. compute effectiveProxy see subgraph EffectiveProxy
3. applyProxy: clears or sets session.setProxy + updates proxyCredentials by username OR password presence
4. sess = session.fromPartition partition
5. if no fp cache entry generateFingerprint containerId deterministic seed
6. buildFingerprintScript fpData string injected later
7. sess.setUserAgent custom userAgent arg OR fpData.uaString
8. new WebContentsView webPreferences session this sess
9. wcPartitionMap set webContents.id → partition
10. ensureFpScript lazy: first main-frame did-start-navigation attaches debugger sends Page.addScriptToEvaluateOnNewDocument OR on failure dom-ready executeJavaScript once
11. build Accept-Language q-value list from fpData.languages store in acceptLanguageByPartition
12. ensureAcceptLanguageHook partition installs onBeforeSendHeaders if first time for partition
13. setWindowOpenHandler always deny non-blank URLs shell receives open-url-in-new-tab with containerId
14. register many webContents listeners all funnel through closure send → tab-updated tabId partial TabInfo
15. context-menu builds dynamic native menu actions run on this view webContents
16. loadURL url — navigation begins
17. tabs.set tabId record return tabId to shell

destroyTab

1. wcPartitionMap delete by webContents id
2. removeChildView best-effort
3. webContents stop destroy
4. tabs.delete tabId
5. if no remaining tab shares containerId fingerprintCache.delete containerId

switch-tab IPC

1. if previous activeTabId: setTabActive false throttle mute removeChildView old WebContentsView
2. lookup next tab
3. setTabActive true addChildView next view resizeView activeTabId = tabId

rotating-proxy-start

1. stopRotatingProxy clears timer state busy map entry
2. rotatingProxyState stores full proxy list and currentIndex
3. applyNextProxy: if swapBusy skip else set busy fetch next proxy by index increment modulo applyProxy reloadProfileTabs finally clear busy
4. setInterval every intervalMinutes

each tick

Every HTTP request from a tab session after hook installed

1. webRequest onBeforeSendHeaders callback
2. inject Accept-Language from map for this partition
3. delete all keys starting with sec-ch-ua case-insensitive
4. Object.assign chromeClientHints frozen object from chromeLikeClientHintHeaders at main startup
5. callback modified headers — aligns TLS Client Hello related hints with spoofed UA

Shell → main command paths

- addTab: createTab then optimistically push TabInfo then switchTab — shell owns tab strip ordering
- saveProfileSettings: mutates profiles state then invoke setProxy and setUserAgent — disk truth is profiles JSON session truth updated async
- ProxyChainSection start: invoke proxy-chain-start main applies localhost chain proxy reloads all tabs of profile
- open-url-in-new-tab listener resolves profileId from IPC arg or active tab or first profile then addTab then navigateTo

Main process — authoritative runtime state

- rotatingProxySwapBusy: profileId → boolean re-entrancy guard
- fingerprintCache: containerId → FingerprintData
- activeTabId: which tab is attached to contentView
- rotatingProxyState: profileId → proxies array + currentIndex
- acceptLanguageByPartition: partition → Accept-Language header value
- proxyChain module: profileId → local net.Server + listening port
- proxyCredentials: partition → username password when proxy needs auth
- wcPartitionMap: webContents.id → partition string for proxy auth
- rotatingProxyTimers: profileId → interval handle
- tabs: Map tabId → Tab id containerId view WebContentsView

Local SOCKS5 hop server proxyChain.ts

1. Browser session points at 127.0.0.1:port
2. net.Server accepts client sockets
3. handleClient registers onData parses SOCKS5 CONNECT
4. SocksClient.createConnection chains through hop list
5. duplex pipe browser socket ↔ final remote

app.on login — proxy authentication

1. if not authInfo.isProxy return
2. lookup partition by webContents.id in wcPartitionMap
3. if creds in proxyCredentials preventDefault callback user pass