Here you go — **M5 in the same style/format as your M2 doc**, prioritized, deduplicated, and kept under 300 steps. It’s a single RTF block you can drop into the repo. Every step creates an artifact used by later steps, so agents can traverse it linearly without loopiness.

{\rtf1\ansi\deff0

\b M5 — TypeScript Core (Builders/Types) — Critical-Path Execution Plan (<=300 steps)\b0\par

\i Scope:\i0 Implement the off-chain core for UC-4: strongly-typed bundle/step schemas, Uniswap V3 route construction, quoting, slippage/twap policy mirroring, ABI serialization, and executor/adapters client shims. Output feeds M6 CLI, M7 tests, M8 observability, M9 gates, M10 docs.\par

\par

\b Phase A — Package scaffolding & build hygiene\b0\par

1. Create branch \ul m5/ts-core\ulnone and open PR "M5: TypeScript Core".\par

2. Create package dir \ul packages/core-exec\ulnone if missing.\par

3. Add \ul packages/core-exec/package.json\ulnone with name \ul @project/core-exec\ulnone, \{"type":"module","private":true\}.\par

4. Add script stubs: \ul build, clean, typecheck, test, lint\ulnone.\par

5. Add \ul tsconfig.json\ulnone extending repo \ul tsconfig.base.json\ulnone with \{"rootDir":"src","outDir":"dist","composite":true\}.\par

6. Add \ul src/index.ts\ulnone exporting nothing yet (compiles empty).\par

7. Wire workspace in \ul pnpm-workspace.yaml\ulnone (ensure package is included).\par

8. Add \ul src/internal/assert.ts\ulnone with tiny helpers: \i assert, invariant, unreachable\i0.\par

9. Add \ul src/internal/units.ts\ulnone (parseUnits/formatUnits via viem or ethers v6; bigint-safe).\par

10. Add \ul src/internal/retry.ts\ulnone with withRetry(fn, {retries, backoffMs}).\par

11. Add \ul src/internal/time.ts\ulnone with nowSec(), toDeadline(+seconds).\par

12. Add \ul src/internal/log.ts\ulnone simple leveled logger honoring \ul LOG\_LEVEL\ulnone.\par

13. Run \ul pnpm -w build\ulnone to confirm base compiles.\par

\par

\b Phase B — ABI & contract bindings\b0\par

14. Create \ul src/abi/\ulnone directory.\par

15. Add canonical ABIs (JSON) for \ul BundleExecutor, SettlementVault, UniswapV3Adapter, QuoterV2\ulnone.\par

16. Export typed interfaces \ul src/abi/index.ts\ulnone with \i as const\i0 ABIs.\par

17. Add \ul src/clients/provider.ts\ulnone to build public RPC client from \ul RPC\_URL\ulnone.\par

18. Add \ul src/clients/executor.ts\ulnone: factory \i makeExecutor(client, address)\i0 exposing view funcs and \i executeBundle\i0.\par

19. Add \ul src/clients/vault.ts\ulnone: \i makeVault\i0 with \i balanceOf, deposit, transferTo\i0.\par

20. Add \ul src/clients/adapter.univ3.ts\ulnone: \i makeUniv3Adapter\i0 with \i expectedOut (optional view), encodeStep convenience.\par

21. Add \ul src/clients/quoter.univ3.ts\ulnone: \i quoteExactInput / quoteExactInputSingle\i0.\par

22. Unit-stub each client with minimal runtime checks (address checksum, ABI presence).\par

23. Export \ul clients\ulnone barrel from \ul src/index.ts\ulnone.\par

\par

\b Phase C — Core types (single source-of-truth)\b0\par

24. Create \ul src/BundleTypes.ts\ulnone with strict TS types (all bigint): \par

25. Define \ul Address, BigintIsh, Hex\ulnone utility types.\par

26. Define \ul StepParams\ulnone \{tokenIn, tokenOut, amountIn, amountOutMin, recipient, deadline, path:Hex, payer?:Address, feeOnTransfer?:boolean\}.\par

27. Define \ul Bundle\ulnone \{venueId:string, steps:Hex[], deadline:bigint, minTotalOut:bigint, tokenIn:Address, tokenOut:Address, recipient:Address\}.\par

28. Define \ul RoutePolicyMirror\ulnone \{maxHops:number, maxSlippageBps:number, deadlineMaxDelaySecs:number, allowedTokens:Address[], allowedFeeTiers:number[]\}.\par

29. Define \ul Quote\ulnone \{amountIn:bigint, amountOut:bigint, hopIndex:number\} and \ul RouteQuote\ulnone array.\par

30. Define \ul BuildContext\ulnone \{chainId:number, quoter:Address, router:Address, policy?:RoutePolicyMirror\}.\par

31. Export everything from \ul src/index.ts\ulnone.\par

\par

\b Phase D — Venues & fee-tier primitives\b0\par

32. Add \ul src/venues/univ3/fees.ts\ulnone with constants FEE\_500=500, FEE\_3000=3000, FEE\_10000=10000.\par

33. Add \ul src/venues/univ3/pools.ts\ulnone helper: \i isFeeAllowed(fee, policy)\i0.\par

34. Add \ul src/venues/univ3/validate.ts\ulnone: input guards (nonzero addresses, \i token[i] != token[i+1]\i0, fees length = tokens.length-1).\par

35. Unit tests for validators.\par

\par

\b Phase E — Uniswap V3 path encoding\b0\par

36. Add \ul src/venues/univ3/path.ts\ulnone with \i encodePath(tokens:Address[], fees:number[]): Hex\i0.\par

37. Implement exact Uniswap format: token(20) + fee(3) ... + token(20).\par

38. Add \i decodePath\i0 (for tests) verifying round-trip.\par

39. Add \i keccakPath\i0 (path hash) utility.\par

40. Unit tests: single-hop and two-hop golden vectors.\par

\par

\b Phase F — Quoter integration & quoting pipeline\b0\par

41. Add \ul src/venues/univ3/quote.ts\ulnone with \i quoteExactInputSingle\i0 and \i quoteExactInputPath\i0.\par

42. For \i quoteExactInputPath\i0, iterate hops: call Quoter for each hop amountOut → feed to next.\par

43. Support \i multicall batching\i0 if Quoter supports; fallback to sequential.\par

44. Add configurable \i slippageBps\i0 default (100 bps) in function options.\par

45. Add retry via withRetry for transient RPC errors.\par

46. Unit tests with mocked client (deterministic outputs per hop).\par

\par

\b Phase G — Policy mirroring (off-chain)\b0\par

47. Add \ul src/policy/mirror.ts\ulnone with functions:\par

48. \i enforceMaxHops(routeTokens[], policy)\i0 → throws if exceed.\par

49. \i enforceAllowedTokens(tokens[], policy)\i0.\par

50. \i enforceAllowedFees(fees[], policy)\i0.\par

51. \i clampSlippage(requestedBps, policy)\i0 → min(requested, policy.maxSlippageBps).\par

52. \i enforceDeadline(deadline, nowSec(), policy.deadlineMaxDelaySecs)\i0.\par

53. Add \i validateBundleShape(bundle)\i0 for final sanity.\par

54. Unit tests for each rule, including negative cases.\par

\par

\b Phase H — Step construction & serialization\b0\par

55. Add \ul src/build/steps.ts\ulnone with \i buildUniv3Steps\i0:\par

56. Inputs: tokens[], fees[], amountIn, recipient, deadline, slippageBps, policy.\par

57. Call quoting pipeline → get per-hop expectedOut.\par

58. Compute \i amountOutMin per hop\i0 using slippageBps.\par

59. Create \i StepParams[]\i0 aligned to adapter expectation.\par

60. Add \i serializeStep(params)\i0 → ABI encode to bytes Hex.\par

61. Add \i serializeSteps(params[])\i0 → Hex[].\par

62. Unit tests: encoded bytes match Solidity decoder (fixture hex).\par

\par

\b Phase I — Bundle composition\b0\par

63. Add \ul src/build/bundle.ts\ulnone with \i buildBundle\i0:\par

64. Inputs: venueId, tokens[], fees[], amountIn, recipient, deadline, slippageBps, policy.\par

65. Compose using buildUniv3Steps → stepsHex.\par

66. Set \i tokenIn=tokens[0], tokenOut=tokens[last]\i0.\par

67. Compute \i minTotalOut\i0 from last hop expectedOut under slippage.\par

68. Return \i Bundle\i0 ready for on-chain call.\par

69. Add \i bundleToCalldata\i0 encoder mirroring Executor ABI.\par

70. Unit tests: bundle integrity (hops consistent, token continuity, deadlines).\par

\par

\b Phase J — Registry/Policy fetch (optional online mirror)\b0\par

71. Add \ul src/clients/registry.ts\ulnone with \i makeRegistry\i0; expose \i getPolicy(venueId)\i0.\par

72. Add helper \i fetchAndMirrorPolicy(venueId)\i0 → RoutePolicyMirror.\par

73. Fallback to local policy if on-chain fetch unavailable (MVP tolerance).\par

74. Unit tests with mock provider returning structured policy.\par

\par

\b Phase K — Executor submission shim\b0\par

75. Add \ul src/execute/submit.ts\ulnone with \i submitBundle(executor, bundle)\i0.\par

76. Populate tx with gas estimation + optional maxFeePerGas override.\par

77. Await receipt and decode \i BundleExecuted\i0 event; return summary.\par

78. Add \i simulateBundle\i0 wrapper that performs eth\_call against Executor (if available) to preflight.\par

79. Unit tests with fake client confirming ABI encoding/decoding paths.\par

\par

\b Phase L — Errors, result types, and mapping to UX\b0\par

80. Add \ul src/errors.ts\ulnone with discriminated unions for BuildError, QuoteError, PolicyError, SubmitError.\par

81. Each has \i code, message, context\i0; codes: E\_MAX\_HOPS, E\_TOKEN\_DISALLOWED, E\_DEADLINE, E\_QUOTE\_RPC, E\_EXECUTOR\_REVERT.\par

82. Add \ul src/format/result.ts\ulnone mappers to concise CLI messages (used later by M6).\par

83. Unit tests for error formatting (stable outputs).\par

\par

\b Phase M — Configuration & addresses loader\b0\par

84. Add \ul src/config/addresses.ts\ulnone that loads \ul config/addresses.json\ulnone and selects by chainId.\par

85. Add \ul src/config/env.ts\ulnone reading RPC\_URL, CHAIN\_ID, optional GAS params.\par

86. Add \ul src/config/index.ts\ulnone aggregator for easy single import.\par

87. Unit tests for config resolution precedence: env → addresses.json.\par

\par

\b Phase N — Public API surface (barrel)\b0\par

88. In \ul src/index.ts\ulnone export:\par

89. Types: \i StepParams, Bundle, RoutePolicyMirror, RouteQuote\i0.\par

90. Builders: \i encodePath, buildUniv3Steps, buildBundle, serializeStep(s), bundleToCalldata\i0.\par

91. Clients: \i makeExecutor, makeUniv3Adapter, makeVault, makeQuoter, makeRegistry\i0.\par

92. Policy mirror: \i fetchAndMirrorPolicy, enforce\*\i0 set.\par

93. Execute: \i submitBundle, simulateBundle\i0.\par

94. Internals: explicitly not exported (private paths maintained).\par

\par

\b Phase O — Golden vectors & cross-language coherence\b0\par

95. Create \ul test/fixtures/univ3.path.json\ulnone with tokens/fees and expected path hex.\par

96. Create \ul test/fixtures/step.hex.json\ulnone matching Solidity decoder fixtures.\par

97. Create \ul test/fixtures/bundle.hex.json\ulnone for end-to-end ABI.\par

98. Add tests to assert equality with these fixtures.\par

\par

\b Phase P — Unit tests (vitest/jest)\b0\par

99. Test \ul path.ts\ulnone: encode/decode; invalid inputs throw.\par

100. Test \ul quote.ts\ulnone: single-hop and two-hop with mocked quoter client.\par

101. Test \ul mirror.ts\ulnone: all policy rules; clamp slippage.\par

102. Test \ul steps.ts\ulnone: minOut derivation; serialization roundtrip.\par

103. Test \ul bundle.ts\ulnone: continuity, minTotalOut, deadline window.\par

104. Test \ul submit.ts\ulnone: calldata payload shape; event decode on fake receipt.\par

105. Ensure coverage > 90% lines for \ul src/build/\*\ulnone and \ul src/venues/univ3/\*\ulnone.\par

\par

\b Phase Q — Integration tests (RPC fork or deterministic mocks)\b0\par

106. Add \ul test/integration/quote.univ3.fork.spec.ts\ulnone that calls real Quoter on fork block.\par

107. Gate by \ul FORK\_URL\ulnone env; otherwise skip with reason.\par

108. Add \ul test/integration/build.bundle.spec.ts\ulnone using fork: build 2-hop bundle WETH→USDC→DAI (or equivalents) at pinned block.\par

109. Assert quoted amounts monotonic and \i minTotalOut <= expectedOut\i0.\par

110. Add \ul test/integration/policy.rejections.spec.ts\ulnone: disallowed token/fee/hops/expired deadline.\par

\par

\b Phase R — Serialization parity tests with Solidity\b0\par

111. Import Solidity decoder test outputs (generated in M7) or add temporary Solidity helper to decode TS-produced bytes.\par

112. Roundtrip: TS→bytes→Solidity decode→TS compare fields.\par

113. Store comparisons in \ul reports/roundtrip.json\ulnone.\par

\par

\b Phase S — Performance & caching\b0\par

114. Add micro-bench \ul test/bench/path.bench.ts\ulnone measuring encodePath for 1–4 hops.\par

115. Add \ul src/cache/memo.ts\ulnone wrapper to memoize identical quote requests for 1 block.\par

116. Wire memoization into \i quoteExactInputPath\i0 behind option flag.\par

117. Document perf notes in code comments (no premature caching by default).\par

\par

\b Phase T — Observability hooks (emit-only; consumed by M8)\b0\par

118. Define \ul src/telemetry/types.ts\ulnone: \i QuoteLatency, BuildLatency, SubmitLatency, ErrorCount\i0.\par

119. Add \ul src/telemetry/emitter.ts\ulnone with pluggable sink (console/NOOP by default).\par

120. Emit events at key points: pre/post quote, build, submit.\par

121. Export \i setTelemetrySink\i0 from index for M8 to configure.\par

\par

\b Phase U — Developer ergonomics\b0\par

122. Add \ul src/format/summary.ts\ulnone that renders human-readable route summary (tokens→fees→amounts) for CLI reuse.\par

123. Add \ul src/ids.ts\ulnone computing deterministic \i bundleId\i0 (keccak of serialized bundle) for logs.\par

124. Provide \ul src/safety/sanitize.ts\ulnone ensuring recipient not zero, amounts > 0.\par

125. Provide \ul src/safety/normalize.ts\ulnone to coerce \i BigintIsh\i0 inputs to bigint.\par

\par

\b Phase V — Public documentation comments (TSDoc)\b0\par

126. Add TSDoc to all exported functions/types with param/returns and error codes.\par

127. Ensure \ul pnpm -w docs:ts\ulnone (if configured) can extract API docs.\par

\par

\b Phase W — CI wiring & quality gates\b0\par

128. Update \ul .github/workflows/ci.yml\ulnone to run \ul pnpm -w --filter @project/core-exec test\ulnone.\par

129. Add coverage upload step to store \ul reports/coverage/ts-core\ulnone.\par

130. Add job \ul ts-core-lint\ulnone to run eslint on \ul packages/core-exec/src\ulnone.\par

131. Add job \ul ts-core-typecheck\ulnone to run tsc -b packages/core-exec.\par

\par

\b Phase X — Address/config interop tests\b0\par

132. Add \ul test/config.addresses.spec.ts\ulnone asserting selection by chainId and presence of router/quoter.\par

133. Add negative tests for missing fields yielding actionable errors.\par

\par

\b Phase Y — Error surfaces & mapping from on-chain custom errors\b0\par

134. Add \ul src/errors/onchain.ts\ulnone mapping revert selectors to friendly codes (if ABIs expose errors).\par

135. In submitBundle, decode revert data (if present) and rethrow as SubmitError with \i selector\i0 field.\par

136. Tests: craft fake revert with known selector and assert mapping.\par

\par

\b Phase Z — End-to-end harness (headless; pre-CLI)\b0\par

137. Add \ul src/run/localExecute.ts\ulnone that:\par

138. Loads config/env; builds a 2-hop route; prints summary; if \ul DRY\_RUN\ulnone false, submits to executor.\par

139. Returns process exit code 0/1; logs bundleId and amounts.\par

140. Add \ul pnpm script\ulnone "core:local:exec": "tsx packages/core-exec/src/run/localExecute.ts".\par

141. Smoke-test against anvil or fork; capture \ul reports/local-exec.out.json\ulnone.\par

\par

\b Phase AA — Stability & race safety\b0\par

142. Ensure \i toDeadline\i0 rounds up to nearest second and guards against clock skew.\par

143. Guard that \i deadline-now <= policy.deadlineMaxDelaySecs\i0 (mirror of on-chain).\par

144. Add \i monotonic tokens\i0 check: tokens[i+1] === next hop input; throw if not.\par

145. Add \i bigint overflow\i0 safety: no Number conversions; tests enforce.\par

\par

\b Phase AB — Versioning & feature flags\b0\par

146. Add \ul src/version.ts\ulnone with \i CORE\_EXEC\_VERSION\i0 (semver string).\par

147. Add \ul src/featureFlags.ts\ulnone enabling "adapter:univ3" (true) and placeholders for future venues.\par

148. Export from index and log in localExecute.\par

\par

\b Phase AC — Developer docs stubs (consumed by M10)\b0\par

149. Create \ul docs/m5/ts-core/README.md\ulnone with API surface summary and examples.\par

150. Add \ul docs/m5/ts-core/examples.md\ulnone showing build+serialize+submit snippets.\par

151. Add \ul docs/m5/ts-core/errors.md\ulnone listing error codes and meanings.\par

\par

\b Phase AD — Cross-package interop hooks (for M6/M7)\b0\par

152. Ensure \ul packages/adapters-evm/src/index.ts\ulnone re-exports selected builder types (read-only) to avoid duplication.\par

153. Publish local type alias \ul @project/types\ulnone (optional) or keep within core-exec (preferred for MVP).\par

\par

\b Phase AE — Deterministic quotes (block pinning option)\b0\par

154. Add option \i blockTag\i0 to Quoter calls; pass through from callers.\par

155. In localExecute, accept \ul --blockTag\ulnone for reproducible outputs.\par

\par

\b Phase AF — Gas & fee awareness (off-chain)\b0\par

156. Add \ul src/gas/price.ts\ulnone to fetch baseFee, priority tip (RPC eth\\_maxPriorityFeePerGas/eth\\_gasPrice fallback).\par

157. Add \ul src/gas/calc.ts\ulnone to estimate cost impact (rough) and annotate route summary.\par

158. Do not alter on-chain behavior; purely informational for CLI.\par

\par

\b Phase AG — Robust JSON schemas\b0\par

159. Add \ul src/schema/bundle.schema.json\ulnone (JSON Schema draft 2020-12).\par

160. Add \ul src/schema/step.schema.json\ulnone.\par

161. Add \ul src/schema/validate.ts\ulnone using ajv to validate external bundle JSON before submit.\par

162. Tests: invalid \i minTotalOut\i0, negative amounts rejected.\par

\par

\b Phase AH — Edge cases & protective checks\b0\par

163. For fee-on-transfer flag, warn in summary that exactOut may differ; keep disabled by default.\par

164. If tokens length=2 and fees length=1 → single-hop fast path.\par

165. Reject empty tokens/fees early with explicit code E\_EMPTY\_ROUTE.\par

166. Clamp \i slippageBps\i0 to [0, 5000] locally regardless of policy; error if request out of bounds.\par

\par

\b Phase AI — Result reporting structs\b0\par

167. Define \ul BuildReport\ulnone \{bundleId, hops, estOut, minOut, deadline, tokens, fees\}.\par

168. Define \ul SubmitReport\ulnone \{txHash, bundleId, gasUsed?, status, totalOut?\}.\par

169. Export report builders in \ul src/format/summary.ts\ulnone.\par

\par

\b Phase AJ — Minimal demo wiring for consumers\b0\par

170. Add \ul examples/m5/build-and-simulate.ts\ulnone using the exported API.\par

171. Add \ul pnpm script\ulnone "core:demo": "tsx packages/core-exec/examples/m5/build-and-simulate.ts".\par

\par

\b Phase AK — Packaging & exports sanity\b0\par

172. Ensure \ul package.json "exports"\ulnone points to \ul ./dist/index.js\ulnone.\par

173. Ensure \ul types\ulnone points to \ul ./dist/index.d.ts\ulnone.\par

174. Build and locally \ul node -e "import('@project/core-exec')"\ulnone to validate ESM export path.\par

\par

\b Phase AL — Logging redaction & safety\b0\par

175. Ensure logs never print private keys or raw RPC URLs; redact with \i redactUrl\i0 helper.\par

176. Add test to assert redactUrl behavior.\par

\par

\b Phase AM — Deterministic hashing helpers\b0\par

177. Add \ul src/crypto/hash.ts\ulnone wrapper using keccak256 on hex inputs; tests for stability.\par

178. Use in \i bundleId\i0 and \i pathHash\i0.\par

\par

\b Phase AN — Compatibility guards\b0\par

179. Validate chainId in BuildContext matches addresses.json selected network; throw E\_CHAIN\_MISMATCH otherwise.\par

180. Expose \i getChainId(client)\i0 utility and assert equality in localExecute.\par

\par

\b Phase AO — DevX polish\b0\par

181. Add \ul pnpm script\ulnone "core:check": "pnpm --filter @project/core-exec lint && pnpm --filter @project/core-exec typecheck && pnpm --filter @project/core-exec test".\par

182. Add badge to root README (M10 will link) indicating M5 API readiness.\par

\par

\b Phase AP — Acceptance tests & checklists\b0\par

183. Write \ul test/acceptance/m5.api.surface.spec.ts\ulnone asserting exported symbols.\par

184. Write \ul test/acceptance/m5.builder.e2e.spec.ts\ulnone (with fork) covering 2-hop happy path to serialized bundle only.\par

185. Add \ul ACCEPTANCE.md\ulnone in package with criteria: unit pass, integration pass (fork optional), coverage >= 85%, serialization parity.\par

\par

\b Phase AQ — CI artifacts & developer hints\b0\par

186. Upload \ul dist/\ulnone and \ul coverage/\ulnone as workflow artifacts for the PR.\par

187. Post a PR comment with bundle/build demo output (from localExecute dry run).\par

\par

\b Phase AR — Final polish & merge\b0\par

188. Run \ul pnpm core:check\ulnone locally; fix any lint/type/format issues.\par

189. Re-run forked integration tests at pinned block; update fixtures if liquidity moved.\par

190. Rebase PR onto main; resolve conflicts.\par

191. Request reviews from Core Exec, Adapters, and Security reviewers; address feedback.\par

192. Squash-merge after two approvals and green CI.\par

193. Tag \ul v0.5.0-ts-core\ulnone (pre-release) and attach \ul reports/roundtrip.json\ulnone.\par

\par

\b Phase AS — Handoffs to downstream milestones\b0\par

194. Open M6 tasks to consume \i summary, report, submitBundle, localExecute\i0 in CLI.\par

195. Open M7 tasks to integrate fixtures and parity tests across Solidity/TS.\par

196. Open M8 tasks to wire telemetry sink into real metrics emitter.\par

197. Open M10 tasks to publish API docs and Quickstart snippets using examples.\par

\par

\b Phase AT — Post-merge hardening (optional, recommended)\b0\par

198. Add fuzz-like randomized tests for \i slippageBps\i0 and hop counts via property testing lib.\par

199. Add concurrency test: parallel quote calls under memoization disabled/enabled.\par

200. Document known limitations in \ul docs/m5/ts-core/README.md\ulnone (e.g., fee-on-transfer accuracy).\par

}

If you want me to **drop in concrete command lines under each step** (e.g., exact viem client calls or ABI-encoder invocations), just say the word and I’ll expand the relevant phases (H–K, P–R) with code snippets that your agents can paste directly.