



Staqtapp-TDS

API Surface Reference

AI-safe calls, stress testing surfaces, Browser/Studio integration, and closed driver-runtime boundaries

Generated from the Driver Studio Stress Scenario Matrix package

Executive API Map

This reference describes the callable TDS surfaces that an AI system, stress harness, browser operations console, or Driver Studio interface should use. It also labels the closed/controlled runtime boundary around the Driver VM.

Primary non-halting result contracts

DriverVMResult is the VM execution envelope. DriverExecutionEvidence is the Runtime Manager gated-execution evidence envelope. DriverFoundryResult is the AI-safe proposal/testing envelope. General storage operations use TDSResult.

AI / Agent proposal or stress tooling



DriverFoundry -> DriverFoundryResult



Runtime Manager -> DriverExecutionEvidence



Driver VM Runtime -> DriverVMResult



Registry / Review Board authority remains outside Studio and AI direct control

Recommended call surface for AI and stress tools

Area	Call	Return	Operational meaning
AI proposal path	DriverFoundry.propose_driver(source)	DriverFoundryResult	AI-safe path: validate, compile, audit, optional fixture tests. Does not approve/sign/activate.
AI validation loop	DriverFoundry.validate_driver(source)	DriverFoundryResult	Parse/validate TDDL without compiling or executing.
AI compile loop	DriverFoundry.compile_driver(source)	DriverFoundryResult	Compile deterministic bytecode package; no trust activation.
AI audit loop	DriverFoundry.audit_driver(package)	DriverFoundryResult	Audit VM contract compatibility before execution.
AI fixture loop	DriverFoundry.test_driver(package, fixtures)	DriverFoundryResult	Safe fixture execution through controlled driver path.
Candidate handoff	DriverFoundry.submit_candidate(package)	DriverFoundryResult	Candidate submission only; Registry still owns approval/signature/activation.
Gated execution	DriverRuntimeManager.execute_package(package, fixtures)	DriverExecutionEvidence	Primary production-facing gate. Returns evidence for accepted/rejected/runtime-faulted outcomes.
Low-level VM runtime	DriverVMRuntime.execute(inputs)	DriverVMResult	Closed/controlled runtime surface; direct use is for internal tests/parity/harness, not open AI authority.
Regression evidence	DriverRegressionHarness.run_package(package, fixtures)	DriverRegressionReport	Deterministic fixture regression evidence for review and stress paths.
Performance evidence	DriverVMPerformanceHarness.run_package(package, fixtures)	DriverVMPerformanceReport	Opt-in, off-hot-path performance/parity evidence; never automatic in normal execution.
Storage read	TSDirectory.read_result(name)	TDSResult	AI-safe non-halting read envelope.
Storage write	TSDirectory.write_result(name, value, ...)	TDSResult	AI-safe non-halting write envelope.

Area	Call	Return	Operational meaning
Storage delete	<code>TSDirectory.delete_result(name)</code>	<code>TDSResult</code>	AI-safe non-halting delete envelope.
Persistence read	<code>TDSReader.read_result(name)</code>	<code>TDSResult</code>	Non-halting mmap persistence read result.
Batch persistence read	<code>TDSReader.read_many_result(names)</code>	<code>TDSResult</code>	Non-halting batch read over a .tds file.
Browser status	<code>AdminControl.status()</code>	<code>dict/status payload</code>	Browser/admin telemetry snapshot. Browser polls /status.json.
Config stage	<code>AdminControl.stage_config(candidate, grant)</code>	<code>admin_result payload</code>	Admin config intent through local-control layer, not raw storage mutation.
Config promote	<code>AdminControl.promote_config(grant)</code>	<code>admin_result payload</code>	Promotes staged config with authorization checks.
Config rollback	<code>AdminControl.rollback_config(grant)</code>	<code>admin_result payload</code>	Rolls back active config through control layer.
Studio shell	<code>StudioQtBridge.load_bundle(bundle)</code>	<code>StudioQtShellState</code>	Read-only/hydrated Driver Studio snapshot entry point.
Studio live runtime	<code>StudioLivePanelRuntime.refresh()</code>	<code>StudioPanelRuntimeState</code>	Bounded event consumption plus panel refresh state; reports retention gaps.
Studio stress harness	<code>StudioOperationalStressHarness.run(...)</code>	<code>StudioOperationalStressReport</code>	Headless Browser/Studio/.tds observer stress evidence; no authority expansion.
Studio stress scenario	<code>StudioOperationalStressHarness.run_scenario(...)</code>	<code>StudioOperationalStressScenarioResult</code>	Runs one named operational stress scenario.
Studio stress matrix	<code>StudioOperationalStressHarness.run_scenario_matrix(...)</code>	<code>StudioOperationalStressScenarioMatrix</code>	Runs the default or selected stress scenario matrix with JSON-safe evidence.
Studio manual preview	<code>StudioManualBuilderUIRuntime.preview_form_payload(payload)</code>	<code>StudioManualBuilderRuntimeState</code>	Deterministic TDDL preview; no authority handoff.
Studio manual proposal	<code>StudioManualBuilderUIRuntime.propose_form_payload(payload)</code>	<code>StudioManualBuilderRuntimeState</code>	Explicit Foundry-routed proposal only.
Export audit	<code>StudioExportAuditConsole.current_state(...)</code>	<code>StudioExportAuditConsoleState</code>	Hash-backed export packet preview; does not sign or authorize.
Export integrity	<code>StudioExportIntegrityWorkflow.current_state(...)</code>	<code>StudioExportIntegrityWorkflowState</code>	Recomputes manifest/packet hashes and emits review-safe readiness gate.

Driver Runtime: Closed / Controlled API Boundary

Yes: the driver runtime should be treated as closed API conditions in the architectural sense. That does not mean there is no callable DriverVMRuntime in the code. It means direct VM execution is not the open AI/public authority path.

Rule	Meaning
Closed means controlled, not absent	DriverVMRuntime.execute() exists, but it is not the preferred AI-facing public authority surface.
AI path goes through Foundry	AI proposals should use DriverFoundryResult-producing calls so validation, compile, audit, and fixture evidence are recorded.
Production execution goes through Runtime Manager	Runtime Manager returns DriverExecutionEvidence and owns policy/registry/signature gates before VM execution.
VM result remains structured	Recoverable VM execution outcomes return DriverVMResult instead of halting host operations.
Studio is observe/intent only	Studio may preview, hydrate, explain, verify, export-prepare, and submit review intent, but cannot approve/sign/activate.

Safe statement

AI systems should normally call DriverFoundry and DriverRuntimeManager surfaces. Direct DriverVMRuntime.execute() is reserved for controlled internal tests, fixture/regression harnesses, performance/parity evidence, and carefully gated runtime-manager use.

Expected result pattern

```
DriverFoundry.propose_driver(source) -> DriverFoundryResult
DriverRuntimeManager.execute_package(...) -> DriverExecutionEvidence
DriverVMRuntime.execute(inputs) -> DriverVMResult
TSDSDirectory.read_result/write_result(...) -> TDSResult
```

The VM HALT instruction is not a host TDS halt. It is a normal bytecode terminator. Recoverable operational failures should return structured results and evidence rather than stopping Staqtapp-TDS operations.

Browser and Driver Studio Connection Model

The TDS Browser and Driver Studio can run at the same time because they consume copied snapshots and bounded event streams. They do not compete for ownership of the .tds file or Registry authority.

Browser operations console

```
TDSFileSystem / TelemetryManager
-> AdminControl.status()
-> AdminPanelServer /status.json
-> Browser dashboard JavaScript polling, default ~2 seconds
```

The Browser is the storage/system telemetry surface. It should poll immutable status payloads, not walk .tds files or acquire storage hot-path locks during refresh.

Driver Studio

```
Driver evidence / runtime snapshots
-> StudioQtBridge
-> StudioCockpitHydrator
-> StudioCockpitEventBridge(max_events=256)
-> StudioLivePanelRuntime
-> PyQt5 panels and widgets
```

The Studio is the driver/evidence/review/export surface. It observes, hydrates, explains, verifies, and prepares intent. It does not approve, sign, activate, mutate Registry trust, or write storage.

.tds file persistence behavior

```
TDSWriter writes *.tds~ shadow
-> fsync shadow
-> atomic os.replace() to final *.tds

TDSReader uses read-only mmap. Existing readers keep a stable view; new readers or reload()
see the replaced file.
```

AI-Safe and Stress Testing Use

AI systems should be given API access that produces evidence and bounded results, not raw authority. Stress tools should exercise the same result-first surfaces used by production paths.

Preferred AI proposal loop

```
foundry = DriverFoundry(policy=DriverFoundryPolicy(...))
result = foundry.propose_driver(tddl_source)

if result.ok:
    # candidate/evidence can move to review path
else:
    # result.faults and result.context guide repair loop
```

Preferred gated execution loop

```
manager = DriverRuntimeManager(policy=RuntimeManagerPolicy(...))
evidence = manager.execute_package(bytecode_package, fixtures=safe_snapshot_fixtures)

# evidence.status communicates policy rejection, package rejection, registry rejection,
# signature rejection, runtime fault, internal error, or successful execution.
```

Preferred performance evidence loop

```
policy = DriverVMPerformancePolicy(...)
harness = DriverVMPerformanceHarness(policy=policy)
report = harness.run_package(bytecode_package, fixtures=safe_snapshot_fixtures)

# This remains opt-in and off the normal driver hot path.
```

Stress testing rule

Stress should amplify evidence generation, fixture coverage, parity checks, event retention behavior, and JSON/signal safety. It should not grant AI or Studio direct Registry authority.

Authority Boundaries

This table is intentionally strict because TDS is designed for AI systems and long-running services. The more capable the UI and AI-facing API becomes, the more explicit the boundary must remain.

Subsystem	Owns	Explicitly does not own
Storage engine	Owns data and persistence	Does not delegate storage sovereignty to Studio, Browser, or AI agents
Driver VM	Owns bytecode execution result shape	Not open AI authority; direct execution is controlled
Runtime Manager	Owns trust/evidence gating before execution	Does not silently authorize through Studio UI state
Foundry	Owns AI-safe proposal, validation, compile, audit, fixture tests	Does not approve, sign, or activate
Registry	Owns approval, signature, activation trust state	Studio/Browsers only observe Registry state
Studio	Owns visibility, workflow, explanation, preview, hydration, review context, export preparation, submitted intent	Cannot approve/reject as authority, sign, activate, mutate Registry, write storage, or bypass policy
Browser	Owns local telemetry/config operations surface	Does not directly walk .tds hot path or own driver trust

Capability Matrix Calls

TDS includes several capability-matrix helpers. These are useful for AI agents, documentation, Studio panes, and stress tooling because they expose what a surface can and cannot do.

Module	Call	Description
staqtapp_tds.drivers.evidence	<code>EvidenceBundleExporter.capability_matrix(self) -> Mapping[str, bool]</code>	Return the export-layer authority boundary for GUI display.
staqtapp_tds.drivers.evidence	<code>evidence_export_capability_matrix(export_format: EvidenceExportFormat str=EvidenceExportFormat.JSON) -> Mapping[str, bool]</code>	Convenience function for displaying evidence-export authority.
staqtapp_tds.drivers.foundry	<code>DriverFoundry.capability_matrix(self) -> Mapping[str, bool]</code>	Return the Foundry authority matrix for AI and Studio surfaces.
staqtapp_tds.drivers.foundry	<code>foundry_capability_matrix(policy: DriverFoundryPolicy None=None) -> Mapping[str, bool]</code>	Convenience function for displaying the AI-safe Foundry authority map.
staqtapp_tds.drivers.performance	<code>DriverVMPPerformanceHarness.capability_matrix(self) -> Mapping[str, bool]</code>	
staqtapp_tds.drivers.performance	<code>driver_vm_performance_capability_matrix(policy: DriverVMPPerformancePolicy None=None, *, native_runner_available: bool=False) -> Mapping[str, bool]</code>	Expose harness boundaries for Admin/Studio documentation.
staqtapp_tds.drivers.regression	<code>DriverRegressionHarness.capability_matrix(self) -> Mapping[str, bool]</code>	Return the harness authority map for Admin and Studio display.
staqtapp_tds.drivers.regression	<code>regression_harness_capability_matrix() -> Mapping[str, bool]</code>	Convenience function for displaying the harness authority map.
staqtapp_tds.drivers.review	<code>DriverBatchReviewBoard.capability_matrix(self) -> Mapping[str, bool]</code>	Return the Admin review authority map for Studio display.
staqtapp_tds.drivers.review	<code>batch_review_capability_matrix(policy: BatchReviewPolicy None=None) -> Mapping[str, bool]</code>	Convenience function for displaying Admin batch review authority.
staqtapp_tds.drivers.runtime_manager	<code>runtime_manager_capability_matrix(policy: RuntimeManagerPolicy None=None) -> Mapping[str, bool]</code>	Display the Runtime Manager authority map for Studio/Admin surfaces.
staqtapp_tds.drivers.studio	<code>DriverStudioReadOnlyConsole.capability_matrix(self) -> Mapping[str, bool]</code>	Return the read-only Studio authority boundary.
staqtapp_tds.drivers.studio	<code>studio_readonly_capability_matrix() -> Mapping[str, bool]</code>	Convenience function for displaying Studio read-only authority.
staqtapp_tds.drivers.studio_actions	<code>DriverStudioAdminReviewActions.capability_matrix(self) -> Mapping[str, bool]</code>	Return the Studio admin-action authority map.
staqtapp_tds.drivers.studio_actions	<code>studio_admin_review_capability_matrix(policy: StudioReviewSubmissionPolicy None=None) -> Mapping[str, bool]</code>	Convenience function for displaying Studio admin-action authority.
staqtapp_tds.drivers.studio_builder	<code>DriverStudioManualProposalBuilder.capability_matrix(self) -> Mapping[str, bool]</code>	Return the workbench authority boundary.
staqtapp_tds.drivers.studio_builder	<code>studio_manual_builder_capability_matrix() -> Mapping[str, bool]</code>	Convenience function for displaying the manual builder boundary.
staqtapp_tds.studio_pyqt5.bridge	<code>StudioQtBridge.capability_matrix(self) -> Mapping[str, bool]</code>	Return the GUI shell authority boundary.
staqtapp_tds.studio_pyqt5.bridge	<code>studio_pyqt5_shell_capability_matrix() -> Mapping[str, bool]</code>	Convenience function for rendering shell authority boundaries.
staqtapp_tds.studio_pyqt5.evidence_timeline	<code>StudioEvidenceTimeline.capability_matrix(self) -> Mapping[str, bool]</code>	
staqtapp_tds.studio_pyqt5.evidence_timeline	<code>studio_evidence_timeline_capability_matrix() -> Mapping[str, bool]</code>	Convenience helper for displaying v3.1.16 timeline boundaries.

Module	Call	Description
staqtapp_tds.studio_pyqt5.export_audit	StudioExportAuditConsole.capability_matrix(self) -> Mapping[str, bool]	
staqtapp_tds.studio_pyqt5.export_audit	studio_export_audit_capability_matrix() -> Mapping[str, bool]	Convenience helper for displaying v3.1.20 export/audit boundaries.
staqtapp_tds.studio_pyqt5.export_integrity_workflow	StudioExportIntegrityWorkflow.capability_matrix(self) -> Mapping[str, bool]	
staqtapp_tds.studio_pyqt5.export_integrity_workflow	studio_export_integrity_workflow_capability_matrix() -> Mapping[str, bool]	Convenience helper for displaying v3.1.20 export-integrity boundaries.
staqtapp_tds.studio_pyqt5.hydratation	StudioCockpitHydrator.capability_matrix(self) -> Mapping[str, bool]	Return the hydratation authority boundary.
staqtapp_tds.studio_pyqt5.hydratation	studio_cockpit_hydratation_capability_matrix() -> Mapping[str, bool]	Convenience helper for displaying hydratation boundaries.
staqtapp_tds.studio_pyqt5.live	StudioCockpitEventBridge.capability_matrix(self) -> Mapping[str, bool]	Return live-bridge capabilities and denied authority.
staqtapp_tds.studio_pyqt5.live	studio_live_event_bridge_capability_matrix() -> Mapping[str, bool]	Convenience helper for displaying v3.1.12 live bridge boundaries.
staqtapp_tds.studio_pyqt5.manual_builder_runtime	StudioManualBuilderUIRuntime.capability_matrix(self) -> Mapping[str, bool]	Return the manual-builder UI runtime boundary.
staqtapp_tds.studio_pyqt5.manual_builder_runtime	studio_manual_builder_ui_runtime_capability_matrix() -> Mapping[str, bool]	Return the v3.1.18 UI runtime capability and boundary matrix.
staqtapp_tds.studio_pyqt5.manual_builder_runtime	studio_qt_visual_quality_capability_matrix() -> Mapping[str, bool]	Return the static visual-review boundary.
staqtapp_tds.studio_pyqt5.operational_stresses	StudioOperationalStressHarness.capability_matrix(self) -> Mapping[str, bool]	Return stress-harness capabilities and denied authority.
staqtapp_tds.studio_pyqt5.operational_stresses	studio_operational_stress_capability_matrix() -> Mapping[str, bool]	Convenience helper for the v3.1.23 operational stress boundary.
staqtapp_tds.studio_pyqt5.review_workflow	StudioReviewWorkflowConsole.capability_matrix(self) -> Mapping[str, bool]	
staqtapp_tds.studio_pyqt5.review_workflow	studio_review_workflow_capability_matrix() -> Mapping[str, bool]	Convenience helper for displaying v3.1.14 workflow boundaries.
staqtapp_tds.studio_pyqt5.risk_intelligence	StudioRiskIntelligenceCards.capability_matrix(self) -> Mapping[str, bool]	
staqtapp_tds.studio_pyqt5.risk_intelligence	studio_risk_intelligence_capability_matrix() -> Mapping[str, bool]	Convenience helper for displaying v3.1.16 Risk Intelligence boundaries.
staqtapp_tds.studio_pyqt5.runtime	StudioLivePanelRuntime.capability_matrix(self) -> Mapping[str, bool]	
staqtapp_tds.studio_pyqt5.runtime	studio_live_panel_runtime_capability_matrix() -> Mapping[str, bool]	Convenience helper for displaying v3.1.13 runtime boundaries.

Full Generated Public Call Index

The following appendix is generated from public classes and functions in the v3.1.23 source tree. It is intentionally broader than the recommended AI-safe API surface above. Some entries are lower-level support calls and should be used only where their subsystem boundary applies.

Core storage/telemetry/result surfaces

staqtaapp_tds.backends.native

Type	Call / Class	Summary
function	<code>load_native_backend(*, shards: int=64, requested: str='auto') -> Any None</code>	Try to load the optional compiled backend without raising.
function	<code>load_native_backend_report(*, shards: int=64, requested: str='auto') -> tuple[Any None, NativeLoadReport]`</code>	Load the native backend and return ``(backend_or_none, NativeLoadReport)``.

staqtaapp_tds.backends.native_index

Type	Call / Class	Summary
class	<code>NativeEntryIndexBackend</code>	EntryIndex backend using a C Swiss-table-inspired bytes->int64 handle map.
method	<code>NativeEntryIndexBackend.put(self, key: str, entry: Any) -> int</code>	
method	<code>NativeEntryIndexBackend.put_many(self, items: List[Tuple[str, Any]]) -> List[int]</code>	Insert many key/value pairs with one native transition.
method	<code>NativeEntryIndexBackend.get_handle(self, key: str) -> int</code>	
method	<code>NativeEntryIndexBackend.get_handles(self, keys: List[str]) -> List[int]</code>	
method	<code>NativeEntryIndexBackend.contains(self, key: str) -> bool</code>	
method	<code>NativeEntryIndexBackend.get(self, key: str, default: Optional[Any]=None) -> Optional[Any]</code>	
method	<code>NativeEntryIndexBackend.get_by_handle(self, handle: int) -> Optional[Any]</code>	
method	<code>NativeEntryIndexBackend.pop(self, key: str, default: Optional[Any]=None) -> Optional[Any]</code>	
method	<code>NativeEntryIndexBackend.pop_many(self, keys: List[str], default: Optional[Any]=None) -> List[Optional[Any]]</code>	
method	<code>NativeEntryIndexBackend.keys(self) -> List[str]</code>	
method	<code>NativeEntryIndexBackend.values(self) -> List[Any]</code>	
method	<code>NativeEntryIndexBackend.items(self) -> List[Tuple[str, Any]]</code>	
method	<code>NativeEntryIndexBackend.stats(self) -> Any</code>	
method	<code>NativeEntryIndexBackend.native_execution_stats(self) -> Dict[str, int bool str]</code>	Return raw native execution counters for dashboard telemetry.

staqtaapp_tds.config

Type	Call / Class	Summary
class	<code>RuntimeConfig</code>	value model fields: config_id, generation, created_at, chunk_bytes, compression, compression_enabled, compression_threshold_bytes, serializer, index_backend, radix_depth, key_id, max_memory_bytes
method	<code>RuntimeConfig.default(config_id: str='rc-default', generation: int=1) -> 'RuntimeConfig'</code>	
method	<code>RuntimeConfig.validate(self) -> None</code>	
method	<code>RuntimeConfig.to_dict(self) -> dict[str, Any]</code>	

Type	Call / Class	Summary
method	<code>RuntimeConfig.from_mapping(cls, data: Mapping[str, Any]) -> 'RuntimeConfig'</code>	
method	<code>RuntimeConfig.fingerprint(self) -> str</code>	
method	<code>RuntimeConfig.next_generation(self, **changes) -> 'RuntimeConfig'</code>	
class	<code>AdminConfig</code>	value model fields: runtime, required_subject, allow_network_panel
method	<code>AdminConfig.from_mapping(cls, data: Mapping[str, Any]) -> 'AdminConfig'</code>	
method	<code>AdminConfig.build_runtime(self) -> tuple[RuntimeConfig, dict[str, Any]]</code>	
class	<code>ConfigRegistry</code>	Thread-safe active/candidate config pointer.
method	<code>ConfigRegistry.active(self) -> RuntimeConfig</code>	
method	<code>ConfigRegistry.stage(self, candidate: RuntimeConfig) -> RuntimeConfig</code>	
method	<code>ConfigRegistry.candidate(self) -> RuntimeConfig None</code>	
method	<code>ConfigRegistry.promote(self, candidate: RuntimeConfig None=None) -> RuntimeConfig</code>	
method	<code>ConfigRegistry.rollback(self) -> RuntimeConfig</code>	
method	<code>ConfigRegistry.snapshot(self) -> dict[str, Any]</code>	

staqtapp_tds.diagnostics

Type	Call / Class	Summary
function	<code>enrich_diagnostic_event(event: Dict[str, int]) -> Dict[str, int str]</code>	Return a UI-friendly copy of a fixed-width native diagnostic event.
function	<code>native_diagnostics_available() -> bool</code>	
function	<code>native_diag_snapshot(*, event_limit: int=32) -> NativeDiagnosticSnapshot</code>	
function	<code>native_diag_reset() -> bool</code>	
function	<code>native_diag_set_enabled(enabled: bool) -> bool</code>	
function	<code>native_diag_mark_degraded(degraded: bool=True) -> bool</code>	
function	<code>native_diag_emit(event: DiagnosticEvent int, value_a: int=0, value_b: int=0) -> bool</code>	
class	<code>DiagnosticEvent</code>	
class	<code>NativeDiagnosticSnapshot</code>	value model fields: schema_version, subsystem, enabled, degraded, sequence, snapshot_build_ns, counters, recent_events
method	<code>NativeDiagnosticSnapshot.from_mapping(cls, data: Dict[str, Any]) -> 'NativeDiagnosticSnapshot'</code>	
method	<code>NativeDiagnosticSnapshot.to_dict(self) -> Dict[str, Any]</code>	

staqtapp_tds.native.manager

Type	Call / Class	Summary
function	<code>get_native_manager() -> NativeEngineManager</code>	
function	<code>native_status_result() -> TDSResult</code>	
function	<code>native_capabilities_result() -> TDSResult</code>	
class	<code>NativeRuntimePlatform</code>	Runtime details used to select and validate native engines.
method	<code>NativeRuntimePlatform.detect(cls) -> 'NativeRuntimePlatform'</code>	
method	<code>NativeRuntimePlatform.as_dict(self) -> Dict[str, Any]</code>	
class	<code>NativeLoadReport</code>	Immutable native engine load/capability report.

Type	Call / Class	Summary
method	<code>NativeLoadReport.as_dict(self) -> Dict[str, Any]</code>	
class	<code>NativeEngineManager</code>	Central non-halting manager for optional compiled TDS engines.
method	<code>NativeEngineManager.inspect_module(self, module_name: str) -> tuple[ModuleType None, NativeLoadReport]</code>	Import a native module once and return a report instead of raising.
method	<code>NativeEngineManager.load_index_backend(self, *, shards: int=64, requested: str='auto', factory: Callable[... , Any] None=None) -> tuple[Any None, NativeLoadReport]</code>	Load the native EntryIndex backend or return a fallback report.
method	<code>NativeEngineManager.status_result(self) -> TDSResult</code>	Return non-halting native manager status for diagnostics.
method	<code>NativeEngineManager.capabilities_result(self) -> TDSResult</code>	Return platform and last-known native capability diagnostics.

staqtapp_tds.result

Type	Call / Class	Summary
function	<code>normalize_result_code(code: TDSResultCode str) -> str</code>	Return the stable string value for a result code.
function	<code>result_info(code: TDSResultCode str) -> TDSResultInfo None</code>	Return registry metadata for *code*, or None when unknown.
function	<code>known_result_codes() -> tuple[str, ...]</code>	Return the stable public TDSResult code catalog.
function	<code>is_known_result_code(code: TDSResultCode str) -> bool</code>	True when *code* is part of the public TDSResult contract.
class	<code>TDSResultCode</code>	Authoritative Staqtapp-TDS result-code enum.
class	<code>TDSResultInfo</code>	Registry metadata for one ``TDSResultCode``.
method	<code>TDSResultInfo.as_dict(self) -> Dict[str, Any]</code>	
class	<code>TDSResult</code>	Standard non-halting result envelope.
method	<code>TDSResult.success(cls, code: TDSResultCode str=TDSResultCode.OK, message: str='', **kw) -> 'TDSResult'</code>	
method	<code>TDSResult.fail(cls, code: TDSResultCode str, message: str, **kw) -> 'TDSResult'</code>	
method	<code>TDSResult.error(cls, code: TDSResultCode str, message: str, **kw) -> 'TDSResult'</code>	Alias for fail() used by hardening code paths.
method	<code>TDSResult.from_exception(cls, code: TDSResultCode str, exc: BaseException, **kw) -> 'TDSResult'</code>	Convert an internal exception into a stable, non-throwing result.
method	<code>TDSResult.known_code(self) -> bool</code>	
method	<code>TDSResult.info(self) -> TDSResultInfo None</code>	
method	<code>TDSResult.as_dict(self) -> Dict[str, Any]</code>	

staqtapp_tds.serialization.manager

Type	Call / Class	Summary
function	<code>get_default_serialization_manager() -> SerializationManager</code>	
class	<code>EncodedPayload</code>	Bytes emitted by a serialization codec plus audit metadata.
class	<code>SerializationCodec</code>	Minimal codec protocol used by the manager and registry.
method	<code>SerializationCodec.can_handle(self, value: Any) -> bool</code>	
method	<code>SerializationCodec.dumps(self, value: Any) -> EncodedPayload</code>	
method	<code>SerializationCodec.loads(self, raw: bytes) -> Any</code>	
class	<code>CodecRegistry</code>	Ordered codec registry.
method	<code>CodecRegistry.register(self, codec: SerializationCodec, *, replace: bool=False) -> None</code>	

Type	Call / Class	Summary
method	CodecRegistry.codec_for_name(self, name: str) -> SerializationCodec	
method	CodecRegistry.codec_for_fmt(self, fmt_id: int) -> Optional[SerializationCodec]	
method	CodecRegistry.infer(self, value: Any) -> SerializationCodec	
method	CodecRegistry.snapshot(self) -> dict[str, Any]	
class	SerializationManager	Single gateway for Python-value serialization and deserialization.
method	SerializationManager.default_registry() -> CodecRegistry	
method	SerializationManager.choose_fmt_id(self, value: Any) -> int	
method	SerializationManager.serialize(self, value: Any, fmt_id: int None=None) -> EncodedPayload	
method	SerializationManager.deserialize(self, raw: bytes, fmt_id: int) -> Any	
method	SerializationManager.snapshot(self) -> dict[str, Any]	

staqtapp_tds.tds_filesystem

Type	Call / Class	Summary
function	encode_header(ts_create: int, ts_mod: int, flags: int, fmt_id: int, subdir_count: int, entry_count: int) -> bytes	
function	decode_header(raw: bytes) -> dict	
function	demo()	
class	FmtID	
class	DirFlags	
class	CompressorRegistry	Codec registry with cached direct fn refs (eliminates dict lookup per call).
method	CompressorRegistry.register(cls, name: str, compress_fn: Callable, decompress_fn: Callable) -> None	
method	CompressorRegistry.compress(cls, data: bytes, codec: str='') -> bytes	
method	CompressorRegistry.decompress(cls, data: bytes, codec: str='') -> bytes	
method	CompressorRegistry.set_default(cls, codec: str) -> None	
class	EntrySchema	value model fields: dtype, shape, python_type
method	EntrySchema.validate(self, value: Any, name: str) -> None	
class	HybridRegistry	Probability-weighted LRU using a fixed numpy structured-array store.
method	HybridRegistry.put(self, key: str, value: Any) -> None	
method	HybridRegistry.get(self, key: str) -> Optional[Any]	
method	HybridRegistry.remove(self, key: str) -> None	
method	HybridRegistry.sorted_keys(self) -> List[str]	
method	HybridRegistry.records(self) -> np.ndarray	Expose structured records for diagnostics/JIT tests without object data.
class	LoopCacheSlot	Stores a value every `cycle` writes.
method	LoopCacheSlot.write(self, value: Any) -> bool	
method	LoopCacheSlot.read(self) -> Any	
class	LoopCacheManager	

Type	Call / Class	Summary
method	LoopCacheManager.register(self, name: str, cycle: int=1) -> LoopCacheSlot	
method	LoopCacheManager.write(self, name: str, value: Any) -> bool	
method	LoopCacheManager.read(self, name: str) -> Any	
method	LoopCacheManager.batch_flush_numpy(self, name: str, arr: np.ndarray, axis: int=0) -> Optional[np.ndarray]	
class	ConcurrencyPool	Singleton thread-pool + async event loop.
method	ConcurrencyPool.acquire(cls) -> 'ConcurrencyPool'	
method	ConcurrencyPool.submit_thread(self, fn, *args, **kwargs)	
method	ConcurrencyPool.run_async(self, coro)	
method	ConcurrencyPool.map_parallel(self, fn, items: list)	
method	ConcurrencyPool.shutdown(self)	
class	SymbolTable	
method	SymbolTable.intern(self, symbol: str) -> int	
method	SymbolTable.swap(self, old_sym: str, new_sym: str, matrix: np.ndarray) -> np.ndarray	
method	SymbolTable.decode_matrix(self, matrix: np.ndarray) -> list	
class	BloomFilter	Bloom filter for string keys.
method	BloomFilter.add(self, key: str) -> None	
class	TDSEntry	value model fields: name, fmt_id, data, ts_written, entry_id, codec, payload_kind, content_hash, raw_size, stored_size, provenance, config_id
method	TDSEntry.serialise(self) -> bytes	
method	TDSEntry.deserialise(cls, name: str, fmt_id: FmtID, buf: bytes, ts_written: int=0, entry_id: str='', codec: str='', provenance: ProvenanceTag str None=None) -> 'TDSEntry'	
class	TDSDirectory	
method	TDSDirectory.set_schema(self, name: str, schema: EntrySchema) -> None	
method	TDSDirectory.header_bytes(self) -> bytes	
method	TDSDirectory.write_entry(self, name: str, value: Any, fmt_id: FmtID=FmtID.PICKLE_OBJ, compress: bool None=False, codec: str='', provenance: ProvenanceTag str None=None) -> 'TDSEntry'	Legacy/raw entry write for internal engine code and migration tools.
method	TDSDirectory.write(self, name: str, value: Any, fmt_id: FmtID=FmtID.PICKLE_OBJ, compress: bool None=False, codec: str='', provenance: ProvenanceTag str None=None) -> TDSResult	Public non-halting write surface: always return TDSResult.
method	TDSDirectory.write_variable(self, name: str, value: Any, *, compress: bool None=None, codec: str='', provenance: ProvenanceTag str None=None) -> 'TDSEntry'	
method	TDSDirectory.write_json(self, name: str, value: Any, *, overwrite: bool=False, compress: bool None=None, codec: str='', provenance: ProvenanceTag str None=None) -> TDSResult	
method	TDSDirectory.write_text(self, name: str, text: str, *, overwrite: bool=False, compress: bool None=None, codec: str='', provenance: ProvenanceTag str None=None) -> TDSResult	Store a whole text file as first-class UTF-8 text.
method	TDSDirectory.write_text_chunked(self, name: str, text: str, *, chunk_size: int=65536, overwrite: bool=False, compress: bool None=None, codec: str='') -> TDSResult	
method	TDSDirectory.read_text(self, name: str) -> str	

Type	Call / Class	Summary
method	<code>TSDirectory.read_text_result(self, name: str) -> TDSResult</code>	AI-safe text read surface: always return TDSResult, never raise.
method	<code>TSDirectory.addvar(self, name: str, data: Any) -> TDSResult</code>	
method	<code>TSDirectory.editvar(self, name: str, data: Any, *, overwrite: bool=True) -> TDSResult</code>	
method	<code>TSDirectory.lockvar(self, name: str, locked: bool=True) -> TDSResult</code>	
method	<code>TSDirectory.unlockvar(self, name: str) -> TDSResult</code>	
method	<code>TSDirectory.stalkvar(self, name: str, data: Any=None) -> TDSResult</code>	
method	<code>TSDirectory.findvar(self, name: str) -> TDSResult</code>	
method	<code>TSDirectory.loadvar(self, name: str) -> Any</code>	
method	<code>TSDirectory.variable_control_snapshot(self) -> dict</code>	
method	<code>TSDirectory.entry_metadata(self, name: str) -> dict</code>	
method	<code>TSDirectory.entry_metadata_result(self, name: str) -> TDSResult</code>	Public non-halting metadata lookup surface.
method	<code>TSDirectory.invariant_report(self) -> dict</code>	
method	<code>TSDirectory.provenance_record(self, name: str) -> np.ndarray</code>	
method	<code>TSDirectory.provenance_record_result(self, name: str) -> TDSResult</code>	Public non-halting provenance record lookup surface.
method	<code>TSDirectory.read_value(self, name: str) -> Any</code>	Legacy/raw value read for internal engine code and migration tools.
method	<code>TSDirectory.read(self, name: str) -> TDSResult</code>	Public non-halting read surface: always return TDSResult.
method	<code>TSDirectory.read_result(self, name: str) -> TDSResult</code>	AI-safe read surface: always return TDSResult, never raise.
method	<code>TSDirectory.write_result(self, name: str, value: Any, fmt_id: FmtID=FmtID.PICKLE_OBJ, compress: bool None=False, codec: str='', provenance: ProvenanceTag str None=None) -> TDSResult</code>	AI-safe write surface: always return TDSResult, never raise.
method	<code>TSDirectory.delete_result(self, name: str) -> TDSResult</code>	AI-safe delete surface: always return TDSResult, never raise.
method	<code>TSDirectory.delete_entry(self, name: str) -> None</code>	Legacy/raw delete for internal engine code and migration tools.
method	<code>TSDirectory.delete(self, name: str) -> TDSResult</code>	Public non-halting delete surface: always return TDSResult.
method	<code>TSDirectory.mkdir(self, name: str, **kwargs) -> 'TSDirectory'</code>	
method	<code>TSDirectory.cd(self, name: str) -> 'TSDirectory'</code>	
method	<code>TSDirectory.ls(self, sort_by_prob: bool=True) -> List[str]</code>	
method	<code>TSDirectory.parallel_read_all(self) -> Dict[str, Any]</code>	Read all entries concurrently.
method	<code>TSDirectory.recursive_join(self, dtype=np.float64, max_depth: int=8) -> np.ndarray</code>	Concatenate all numpy array entries in the subtree.
method	<code>TSDirectory.build_offset_index(self) -> np.ndarray</code>	
method	<code>TSDirectory.to_bytes(self) -> bytes</code>	serialise payloads in parallel, then join.
method	<code>TSDirectory.path(self) -> str</code>	
method	<code>TSDirectory.telemetry_snapshot(self) -> dict</code>	
method	<code>TSDirectory.capability_names(self) -> List[str]</code>	

Type	Call / Class	Summary
method	<code>TSDirectory.is_reserved_namespace(self, name: str) -> bool</code>	
method	<code>TSDirectory.reserved_namespace_names(self) -> List[str]</code>	
method	<code>TSDirectory.srz_record(self) -> np.ndarray</code>	
class	<code>TDSFileSystem</code>	---- Usage ----
method	<code>TDSFileSystem.observation_snapshot(self, *, force: bool=False) -> Dict[str, object]</code>	Return the cached dashboard/observability snapshot.
method	<code>TDSFileSystem.resolve(self, path: str) -> TSDirectory</code>	
method	<code>TDSFileSystem.resolve_radix(self, path: str) -> TSDirectory</code>	
method	<code>TDSFileSystem.makedirs(self, path: str, **kwargs) -> TSDirectory</code>	
method	<code>TDSFileSystem.parallel_batch_write(self, writes: List[Tuple[str, str, Any]]) -> None</code>	
method	<code>TDSFileSystem.snapshot_headers(self) -> Dict[str, dict]</code>	
method	<code>TDSFileSystem.telemetry_snapshot(self) -> Dict[str, dict]</code>	
method	<code>TDSFileSystem.capability_snapshot(self) -> Dict[str, List[str]]</code>	
class	<code>WriteAheadLog</code>	Append-only WAL for TSDirectory writes.
method	<code>WriteAheadLog.append(self, entry: 'TDSEntry') -> None</code>	
method	<code>WriteAheadLog.checkpoint(self) -> None</code>	
method	<code>WriteAheadLog.replay(self, directory: 'TSDirectory') -> int</code>	
method	<code>WriteAheadLog.close(self) -> None</code>	

stagtapp_tds.tds_json

Type	Call / Class	Summary
function	<code>codec_stats() -> JsonCodecStats</code>	
function	<code>reset_codec_stats() -> None</code>	
function	<code>preferred_loads_backend() -> str</code>	
function	<code>preferred_dumps_backend() -> str</code>	
function	<code>loads_fast(raw: bytes bytearray memoryview str) -> Tuple[Any, str]</code>	Parse JSON using the fastest available centralized backend.
function	<code>loads_strict(raw: bytes bytearray memoryview str, *, expected_type: type tuple[type, ...] None=None) -> Tuple[Any, str]</code>	
function	<code>loads_manifest(raw: bytes bytearray memoryview str) -> Tuple[dict[str, Any], str]</code>	
function	<code>loads_snapshot(raw: bytes bytearray memoryview str) -> Tuple[dict[str, Any], str]</code>	
function	<code>loads_policy(raw: bytes bytearray memoryview str) -> Tuple[dict[str, Any], str]</code>	
function	<code>dumps_canonical(value: Any) -> Tuple[bytes, str]</code>	Emit deterministic compact JSON bytes through the centralized codec.
function	<code>dumps_status(value: Mapping[str, Any]) -> Tuple[bytes, str, int]</code>	Emit compact status/browser telemetry JSON for polling endpoints.
function	<code>dumps_pretty(value: Any) -> Tuple[str, str]</code>	
function	<code>dumps_snapshot(value: Mapping[str, Any]) -> Tuple[bytes, str, int]</code>	
function	<code>backend_probe() -> JsonBackendInfo</code>	

Type	Call / Class	Summary
class	JsonBackendInfo	value model fields: loads_backend, dumps_backend, parse_ns, dump_ns, simdjson_available, orjson_available
class	JsonCodecStats	value model fields: loads_calls, dumps_calls, parse_ns, dump_ns, simdjson_reads, stdlib_reads, orjson_writes, stdlib_writes, parse_failovers, dump_failovers
method	JsonCodecStats.avg_parse_ns(self) -> int	
method	JsonCodecStats.avg_dump_ns(self) -> int	
method	JsonCodecStats.to_dict(self) -> dict[str, int bool str]	

staqtapp_tds.tds_persistence

Type	Call / Class	Summary
class	SlotRecord	value model fields: name, name_hash, offset, length, fmt_id
class	SlotIndex	O(1) slot lookup via dict primary path.
method	SlotIndex.add(self, record: SlotRecord) -> None	
method	SlotIndex.lookup(self, name: str) -> Optional[SlotRecord]	
method	SlotIndex.all_records(self) -> List[SlotRecord]	
method	SlotIndex.to_bytes(self) -> bytes	JIT kernel fills fixed 24-byte headers in one pass;
method	SlotIndex.from_bytes(cls, buf: bytes, slot_count: int) -> 'SlotIndex'	uses memoryview to avoid redundant byte copies on each record.
class	TDSReader	Mmap random-access reader for a .tds file.
method	TDSReader.reload(self) -> None	
method	TDSReader.read(self, name: str) -> Any	
method	TDSReader.read_raw(self, name: str) -> bytes	
method	TDSReader.read_result(self, name: str) -> TDSResult	Public non-halting persistence read surface: always return TDSResult.
method	TDSReader.read_many(self, names: List[str]) -> Dict[str, Any]	
method	TDSReader.read_many_result(self, names: List[str]) -> TDSResult	Public non-halting batch persistence read surface.
method	TDSReader.keys(self) -> List[str]	
method	TDSReader.close(self) -> None	
class	TDSWriter	Atomic writer: shadow file -> fsync -> rename.
method	TDSWriter.write(self, directory: TDSDirectory, recurse: bool=True) -> int	
method	TDSWriter.write_parallel(self, directory: TDSDirectory) -> int	
class	TDSPersistence	Mounts a TDSFileSystem to a real directory.
method	TDSPersistence.flush_node(self, node: TDSDirectory, parallel: bool=False) -> Tuple[str, int]	
method	TDSPersistence.flush(self, fs: TDSFileSystem, parallel_nodes: bool=True) -> Dict[str, int]	
method	TDSPersistence.load_node(self, tds_path, into: Optional[TDSDirectory]=None) -> TDSDirectory	
method	TDSPersistence.close_reader(self, tds_path) -> None	
method	TDSPersistence.mount(self, fs: TDSFileSystem) -> None	
method	TDSPersistence.unmount(self) -> Dict[str, int]	
method	TDSPersistence.open_readers(self, paths) -> List[TDSReader]	
class	ParallelFlusher	Schedules concurrent flush of multiple TDSDirectory nodes.

Type	Call / Class	Summary
method	<code>ParallelFlusher.enqueue(self, node: TDSDirectory) -> None</code>	
method	<code>ParallelFlusher.flush_all(self, parallel: bool=True) -> Dict[str, int]</code>	

staqtapp_tds.tds_pickle

Type	Call / Class	Summary
function	<code>loads_pickle(raw: bytes, policy: PicklePolicy None=None) -> Any</code>	Deserialize a pickle payload under the configured TDS policy.
function	<code>dumps_pickle(value: Any, policy: PicklePolicy None=None) -> bytes</code>	Serialize a Python object with a TDS envelope and optional safety validation.
function	<code>pickle_policy_snapshot(policy: PicklePolicy None=None) -> dict[str, Any]</code>	
class	<code>PicklePolicyError</code>	Raised when a pickle payload violates the TDS pickle policy.
class	<code>PicklePolicy</code>	Policy for the Python-object compatibility lane.
method	<code>PicklePolicy.from_env(cls) -> 'PicklePolicy'</code>	
method	<code>PicklePolicy.unsafe(self) -> bool</code>	
class	<code>RestrictedUnpickler</code>	Unpickler that only resolves explicitly approved value classes.
method	<code>RestrictedUnpickler.find_class(self, module: str, name: str) -> Any</code>	

staqtapp_tds.telemetry

Type	Call / Class	Summary
class	<code>TelemetryMode</code>	
class	<code>TelemetryLevel</code>	Snapshot detail levels for the one-way observation layer.
method	<code>TelemetryLevel.coerce(cls, value: 'TelemetryLevel str int') -> 'TelemetryLevel'</code>	
class	<code>TraceRecord</code>	value model fields: timestamp_ns, route_id, elapsed_ns, result_code, bucket
class	<code>DirectoryTelemetry</code>	value model fields: mode, latency_policy, trace_window
method	<code>DirectoryTelemetry.enabled(self) -> bool</code>	
method	<code>DirectoryTelemetry.start(self) -> int</code>	
method	<code>DirectoryTelemetry.record_lookup(self, elapsed_ns: int, *, hit: bool, cold: bool=False, error_code: int=0, route_id: int=0) -> None</code>	
method	<code>DirectoryTelemetry.snapshot(self) -> Dict[str, int str]</code>	
method	<code>DirectoryTelemetry.trace_snapshot(self) -> List[Dict[str, int]]</code>	
method	<code>DirectoryTelemetry.restore_snapshot(self, data: Dict[str, int str]) -> None</code>	
class	<code>TelemetrySnapshot</code>	Immutable dashboard-facing snapshot.
method	<code>TelemetrySnapshot.to_dict(self) -> Dict[str, object]</code>	
class	<code>TelemetryManager</code>	Low-interference observation manager for v2.3.
method	<code>TelemetryManager.set_level(self, level: TelemetryLevel str int) -> None</code>	
method	<code>TelemetryManager.latest_snapshot(self) -> Dict[str, object]</code>	Return the last immutable published snapshot, building one if needed.
method	<code>TelemetryManager.publish_snapshot(self, snapshot: Dict[str, object], *, duration_ns: int=0) -> None</code>	

Type	Call / Class	Summary
method	TelemetryManager.health_state(self, *, snapshot_age_seconds: float None=None, counters: Dict[str, int] None=None, components: Dict[str, Dict[str, object]] None=None) -> Dict[str, object]	
method	TelemetryManager.register_sampler(self, name: str, sampler) -> None	
method	TelemetryManager.set_component(self, name: str, *, status: str='healthy', **fields) -> None	
method	TelemetryManager.incr(self, key: str, amount: int=1) -> None	
method	TelemetryManager.add_bytes(self, *, raw: int=0, stored: int=0) -> None	
method	TelemetryManager.record_timer(self, key: str, elapsed_ns: int) -> None	
method	TelemetryManager.record_execution(self, *, python_ns: int=0, native_ns: int=0, gil_released_ns: int=0, transitions: int=0, native_ops: int=0, python_ops: int=0, batch_ops: int=0) -> None	Record execution-mode telemetry for performance engineering.
method	TelemetryManager.merge_native_execution_stats(self, stats: Dict[str, object]) -> None	Merge optional native-extension execution counters into telemetry.
method	TelemetryManager.record_read(self, elapsed_ns: int, *, hit: bool=True, backend: str='') -> None	
method	TelemetryManager.record_write(self, elapsed_ns: int, *, raw_size: int=0, stored_size: int=0, backend: str='') -> None	
method	TelemetryManager.record_delete(self) -> None	
method	TelemetryManager.record_error(self) -> None	
method	TelemetryManager.record_chunk(self, count: int, elapsed_ns: int=0) -> None	
method	TelemetryManager.record_telemetry_drop(self, amount: int=1) -> None	
method	TelemetryManager.record_telemetry_skip(self, amount: int=1) -> None	
method	TelemetryManager.record_chunk_transition(self, state: str, amount: int=1) -> None	
method	TelemetryManager.record_json(self, *, parse_ns: int=0, serialize_ns: int=0, backend: str='') -> None	Record JSON boundary cost without retaining payloads or parser state.
method	TelemetryManager.record_spiral_event(self, kind: str, amount: int=1) -> None	Record optional Spiral-compatible pipeline activity.
method	TelemetryManager.snapshot(self, *, force: bool=False) -> Dict[str, object]	
class	TelemetryPublisherThread	Dedicated one-way snapshot publisher for hardening phase 1.
method	TelemetryPublisherThread.running(self) -> bool	
method	TelemetryPublisherThread.start(self) -> 'TelemetryPublisherThread'	
method	TelemetryPublisherThread.stop(self, timeout: float None=2.0) -> None	

Browser/Admin operations console

staqtapp_tds.admin.audit

Type	Call / Class	Summary
class	AuditEvent	value model fields: action, subject, config_id, generation, ts, detail
class	AuditLog	
method	AuditLog.record(self, event: AuditEvent) -> None	

Type	Call / Class	Summary
method	<code>AuditLog.entries(self) -> list[dict[str, Any]]</code>	

staqtapp_tds.admin.auth

Type	Call / Class	Summary
class	<code>ConfigGrant</code>	value model fields: subject, action, issued_at, expires_at, nonce, signature
method	<code>ConfigGrant.is_valid(self, secret: str, action: str None=None) -> bool</code>	
class	<code>LocalAuthProvider</code>	
method	<code>LocalAuthProvider.assert_safe_for_bind(self, host: str) -> None</code>	
method	<code>LocalAuthProvider.issue(self, action: str, *, subject: str None=None, ttl_seconds: int=300) -> ConfigGrant</code>	
method	<code>LocalAuthProvider.verify(self, grant: ConfigGrant, action: str) -> None</code>	

staqtapp_tds.admin.console

Type	Call / Class	Summary
function	<code>main(argv=None)</code>	

staqtapp_tds.admin.control

Type	Call / Class	Summary
class	<code>AdminControl</code>	
method	<code>AdminControl.status(self) -> dict[str, Any]</code>	
method	<code>AdminControl.stage_config(self, candidate: RuntimeConfig, grant: ConfigGrant) -> RuntimeConfig</code>	
method	<code>AdminControl.promote_config(self, grant: ConfigGrant) -> RuntimeConfig</code>	
method	<code>AdminControl.rollback_config(self, grant: ConfigGrant) -> RuntimeConfig</code>	

staqtapp_tds.admin.panel

Type	Call / Class	Summary
function	<code>render_dashboard_html(*, version: str=__version__, refresh_seconds: float=PANEL_REFRESH_SECONDS, csrf_token: str='') -> str</code>	Render the packaged dashboard shell.
class	<code>AdminPanelServer</code>	Optional localhost browser panel.
method	<code>AdminPanelServer.make_handler(self)</code>	
method	<code>AdminPanelServer.serve_forever(self)</code>	

staqtapp_tds.admin.spiral_rank

Type	Call / Class	Summary
function	<code>spiral_rank_snapshot_from(source: Any None) -> dict[str, Any]</code>	Best-effort extraction of Spiral Rank telemetry from an arbitrary source.
class	<code>SpiralRankTelemetry</code>	Loss-tolerant observer cache for Spiral Rank browser telemetry.
method	<code>SpiralRankTelemetry.observe_run(self, run: SpiralRankRun) -> None</code>	Record a completed rank run for cached browser feedback.
method	<code>SpiralRankTelemetry.observe_stats(self, stats: SpiralRankStats, results: Iterable[SpiralRankRecord] None=None) -> None</code>	Record stats with optional result rows as a synthetic run bundle.
method	<code>SpiralRankTelemetry.last_stats(self) -> SpiralRankStats None</code>	

Type	Call / Class	Summary
method	<code>SpiralRankTelemetry.snapshot(self) -> dict[str, Any]</code>	Return a JSON-safe immutable snapshot for the admin status payload.

Driver foundation and packaging

staqtapp_tds.drivers.audit

Type	Call / Class	Summary
function	<code>vm_contract_table() -> Mapping[str, Mapping[str, Any]]</code>	Return VM-readiness metadata used by tests, Builder and future Studio.
function	<code>audit_vm_contract(package: BytecodePackage) -> None</code>	Fail-closed audit before a bytecode package may be loaded by the VM skeleton.
class	<code>VMInstructionContract</code>	value model fields: name, opcode, cost, deterministic, allowed_driver_classes, required_capability, may_branch, may_call_adapter, may_propose_evolution
method	<code>VMInstructionContract.to_dict(self) -> dict[str, Any]</code>	

staqtapp_tds.drivers.bytecode

Type	Call / Class	Summary
function	<code>compile_tddl(source_or_program: str TDDLProgram) -> BytecodePackage</code>	Compile TDDL source/program into deterministic non-executing bytecode.
function	<code>compile_program(program: TDDLProgram, *, source_material: str Mapping[str, Any] None=None) -> BytecodePackage</code>	Compile an already-validated TDDL program to a bytecode package.
function	<code>validate_bytecode_package(package: BytecodePackage) -> None</code>	Validate bytecode artifact integrity without executing it.
function	<code>decompile_to_ir(package: BytecodePackage) -> TDDLProgram</code>	Round-trip bytecode package back to readable non-executing TDDL IR.
function	<code>opcode_table() -> Mapping[str, Mapping[str, Any]]</code>	Return stable opcode metadata for future native VM and Studio display.
class	<code>BytecodeOpcode</code>	Stable opcode mapping for the first non-executing driver bytecode set.
class	<code>BytecodeInstruction</code>	Portable bytecode instruction.
method	<code>BytecodeInstruction.to_dict(self) -> dict[str, Any]</code>	
method	<code>BytecodeInstruction.from_dict(cls, data: Mapping[str, Any]) -> 'BytecodeInstruction'</code>	
class	<code>BytecodePackage</code>	Non-executing compiled driver artifact for registry/builder validation.
method	<code>BytecodePackage.to_dict(self) -> dict[str, Any]</code>	
method	<code>BytecodePackage.to_bytes(self) -> bytes</code>	Return canonical package bytes including the package hash.
method	<code>BytecodePackage.to_unsigned_dict(self) -> dict[str, Any]</code>	
method	<code>BytecodePackage.verify_hash(self) -> bool</code>	
method	<code>BytecodePackage.from_bytes(cls, payload: bytes) -> 'BytecodePackage'</code>	

staqtapp_tds.drivers.manifest

Type	Call / Class	Summary
function	<code>validate_manifest(manifest: DriverManifest) -> None</code>	Validate the draft driver manifest contract.
class	<code>DriverSafety</code>	Declared driver safety class.
class	<code>DriverManifest</code>	Minimal typed manifest for future compiled ``.tdd`` packages.
method	<code>DriverManifest.canonical_payload(self) -> bytes</code>	Return deterministic bytes suitable for signing or hashing.

Type	Call / Class	Summary
method	<code>DriverManifest.from_mapping(cls, data: Mapping[str, object]) -> 'DriverManifest'</code>	Build a manifest from a mapping and normalize list-like fields.

staqtapp_tds.drivers.tddl

Type	Call / Class	Summary
function	<code>instruction_specs() -> Mapping[str, InstructionSpec]</code>	Return the self-describing instruction metadata table for Studio/Builder use.
function	<code>parse_tddl(source: str) -> TDDLProgram</code>	Parse strict TDDL source into a non-executing intermediate representation.
function	<code>validate_tddl(program: TDDLProgram) -> None</code>	Validate TDDL syntax contracts without executing driver behavior.
class	<code>TDDLValidationError</code>	Raised when TDDL source fails closed during parsing or validation.
class	<code>InstructionName</code>	
class	<code>TDDLInstruction</code>	value model fields: name, operands, line
class	<code>TDDLProgram</code>	value model fields: driver_id, version, manifest, capabilities, adapters, limits, instructions, evolution
method	<code>TDDLProgram.instruction_names(self) -> tuple[str, ...]</code>	
class	<code>InstructionSpec</code>	value model fields: name, required, optional, allowed_values
method	<code>InstructionSpec.allowed(self) -> frozenset[str]</code>	

staqtapp_tds.drivers.trace

Type	Call / Class	Summary
function	<code>rank_traces(traces: list[TraceEvidence]) -> list[TraceEvidence]</code>	Return traces in deterministic best-first order.
class	<code>TraceEvidence</code>	value model fields: driver_id, path, semantic_score, manifest_score, extraction_score, lineage_trust
method	<code>TraceEvidence.rank_score(self) -> float</code>	

Foundry - AI-safe proposal/testing

staqtapp_tds.drivers.foundry

Type	Call / Class	Summary
function	<code>foundry_capability_matrix(policy: DriverFoundryPolicy None=None) -> Mapping[str, bool]</code>	Convenience function for displaying the AI-safe Foundry authority map.
class	<code>FoundryStage</code>	Driver Foundry pipeline stages exposed to AI/Studio callers.
class	<code>FoundryStatus</code>	Non-halting status values for Driver Foundry operations.
class	<code>FoundryFault</code>	Structured Foundry fault for repair loops and Studio display.
class	<code>DriverFoundryContext</code>	Compact context attached to every Driver Foundry result.
class	<code>DriverFoundryResult</code>	Result-first envelope for Driver Foundry API calls.
class	<code>DriverFoundryPolicy</code>	Safe authority policy for AI-facing Driver Foundry calls.
class	<code>DriverFoundry</code>	AI-safe build/test/candidate API for TDS drivers.
method	<code>DriverFoundry.capability_matrix(self) -> Mapping[str, bool]</code>	Return the Foundry authority matrix for AI and Studio surfaces.
method	<code>DriverFoundry.propose_driver(self, source: str, *, fixtures: Mapping[str, Any] None=None) -> DriverFoundryResult</code>	Run the AI proposal path: validate, compile, audit, and optionally test.
method	<code>DriverFoundry.validate_driver(self, source: str) -> DriverFoundryResult</code>	Validate TDDL source without compiling or executing it.
method	<code>DriverFoundry.compile_driver(self, source: str) -> DriverFoundryResult</code>	Compile TDDL source to a deterministic bytecode package.
method	<code>DriverFoundry.audit_driver(self, package: BytecodePackage) -> DriverFoundryResult</code>	Audit a compiled package against VM contract rules.

Type	Call / Class	Summary
method	<code>DriverFoundry.test_driver(self, package: BytecodePackage, fixtures: Mapping[str, Any]) -> DriverFoundryResult</code>	Execute a package through DriverVMRuntime against safe snapshot fixtures.
method	<code>DriverFoundry.submit_candidate(self, package: BytecodePackage, *, registry: DriverRegistry, vm_result: DriverVMResult None=None) -> DriverFoundryResult</code>	Submit an audited/tested package as a registry candidate only.

Runtime Manager - gated execution

staqtapp_tds.drivers.runtime_manager

Type	Call / Class	Summary
function	<code>runtime_manager_capability_matrix(policy: RuntimeManagerPolicy None=None) -> Mapping[str, bool]</code>	Display the Runtime Manager authority map for Studio/Admin surfaces.
class	<code>RuntimeManagerStatus</code>	Non-halting status values for managed Driver VM execution.
class	<code>RuntimeManagerFault</code>	Structured manager-level fault for approval and Studio surfaces.
class	<code>RuntimeManagerPolicy</code>	Policy gates enforced before DriverVMRuntime execution.
class	<code>DriverExecutionEvidence</code>	Approval-grade evidence bundle for one managed VM execution.
class	<code>DriverRuntimeManager</code>	Production-facing gate for Driver VM execution.
method	<code>DriverRuntimeManager.execute_package(self, package: BytecodePackage, fixtures: Mapping[str, Any] None=*, registry: DriverRegistry None=None) -> DriverExecutionEvidence</code>	Validate, gate, execute, and return evidence without expected raises.

Driver VM - controlled runtime

staqtapp_tds.drivers.vm

Type	Call / Class	Summary
class	<code>VMState</code>	
class	<code>VMStatus</code>	Structured execution status for non-halting Driver VM calls.
class	<code>VMLoadedPackage</code>	value model fields: driver_id, driver_version, driver_class, instruction_count, package_hash, cost_budget
class	<code>VMFault</code>	One structured runtime fault suitable for Studio and telemetry panels.
class	<code>DriverVMContext</code>	Compact execution context captured on success and failure.
class	<code>DriverVMResult</code>	Result-first Driver VM execution envelope.
class	<code>DriverVMSkeleton</code>	Fail-closed VM loader shell.
method	<code>DriverVMSkeleton.load(self, package: BytecodePackage) -> VMLoadedPackage</code>	Validate and load a package.
method	<code>DriverVMSkeleton.execute(self, _inputs: Mapping[str, Any] None=None) -> DriverVMResult</code>	Fail closed: skeleton intentionally does not execute driver bytecode.
class	<code>DriverVMRuntime</code>	Deterministic runtime for the safe search/extraction opcode set.
method	<code>DriverVMRuntime.execute(self, inputs: Mapping[str, Any] None=None) -> DriverVMResult</code>	

Stress, regression, performance evidence

staqtapp_tds.drivers.performance

Type	Call / Class	Summary
function	<code>driver_vm_performance_enabled(env: Mapping[str, str] None=None) -> bool</code>	Return whether environment-gated benchmark tooling is explicitly enabled.

Type	Call / Class	Summary
function	<code>driver_vm_performance_capability_matrix(policy : DriverVMPPerformancePolicy None=None, *, native_runner_available: bool=False) -> Mapping[str, bool]</code>	Expose harness boundaries for Admin/Studio documentation.
class	<code>DriverVMPPerformanceStatus</code>	Status for opt-in performance evidence generation.
class	<code>DriverVMPPerformanceBackend</code>	Known execution targets for performance evidence.
class	<code>DriverVMPPerformancePolicy</code>	Configuration for controlled VM performance evidence runs.
class	<code>DriverVMPPerformanceRun</code>	One timed VM/backend execution.
method	<code>DriverVMPPerformanceRun.elapsed_ms(self) -> float</code>	
method	<code>DriverVMPPerformanceRun.records_per_second(self) -> float</code>	
method	<code>DriverVMPPerformanceRun.emitted_per_second(self) -> float</code>	
method	<code>DriverVMPPerformanceRun.cost_per_second(self) -> float</code>	
method	<code>DriverVMPPerformanceRun.as_row(self) -> Mapping[str, Any]</code>	
class	<code>DriverVMPPerformanceSummary</code>	Aggregate timing/cost summary for one backend.
method	<code>DriverVMPPerformanceSummary.as_row(self) -> Mapping[str, Any]</code>	
class	<code>DriverVMPPerformanceComparison</code>	Parity/speed comparison between two backend summaries.
method	<code>DriverVMPPerformanceComparison.as_row(self) -> Mapping[str, Any]</code>	
class	<code>DriverVMPPerformanceReport</code>	Evidence report emitted by the opt-in performance harness.
method	<code>DriverVMPPerformanceReport.summary(self, backend: DriverVMPPerformanceBackend str) -> DriverVMPPerformanceSummary</code>	
method	<code>DriverVMPPerformanceReport.signal_payload(self) -> Mapping[str, Any]</code>	Return compact JSON-friendly evidence for future Studio panels.
class	<code>DriverVMPPerformanceHarness</code>	Explicit, off-hot-path performance/parity evidence generator.
method	<code>DriverVMPPerformanceHarness.capability_matrix(self) -> Mapping[str, bool]</code>	
method	<code>DriverVMPPerformanceHarness.run_package(self, package: BytecodePackage, fixtures: Mapping[str, Any] None) -> DriverVMPPerformanceReport</code>	Run opt-in performance evidence for one package and immutable snapshot.

staqtapp_tds.drivers.regression

Type	Call / Class	Summary
function	<code>regression_harness_capability_matrix() -> Mapping[str, bool]</code>	Convenience function for displaying the harness authority map.
function	<code>runtime_fixture_hash(fixtures: Mapping[str, Any]) -> str</code>	Hash fixtures using the same canonical JSON discipline as evidence.
class	<code>RegressionStatus</code>	Top-level status for a regression harness run.
class	<code>DriverFixtureCase</code>	One named fixture and its expected Runtime Manager evidence shape.
class	<code>RegressionMismatch</code>	One deterministic expectation mismatch.
class	<code>DriverRegressionResult</code>	Harness result for one fixture case.
class	<code>DriverRegressionReport</code>	Deterministic report across all fixture cases for one package.
method	<code>DriverRegressionReport.failed_cases(self) -> tuple[str, ...]</code>	
method	<code>DriverRegressionReport.passed_cases(self) -> tuple[str, ...]</code>	
class	<code>DriverRegressionHarness</code>	Run approval-grade Runtime Manager evidence across fixture cases.
method	<code>DriverRegressionHarness.capability_matrix(self) -> Mapping[str, bool]</code>	Return the harness authority map for Admin and Studio display.

Type	Call / Class	Summary
method	<code>DriverRegressionHarness.run_package(self, package: BytecodePackage, cases: Sequence[DriverFixtureCase Mapping[str, Any]], *, registry: DriverRegistry None=None) -> DriverRegressionReport</code>	Run a package against named fixtures without raising for expected failures.

Driver trust/evidence/review surfaces

staqtapp_tds.drivers.evidence

Type	Call / Class	Summary
function	<code>evidence_export_capability_matrix(export_format: EvidenceExportFormat str=EvidenceExportFormat.JSON) -> Mapping[str, bool]</code>	Convenience function for displaying evidence-export authority.
class	<code>EvidenceExportFormat</code>	Supported read-only export formats.
class	<code>EvidenceIntegrityStatus</code>	Integrity status for deterministic evidence bundles.
class	<code>AuditTrailStatus</code>	Completeness status for exported audit trails.
class	<code>EvidenceBundleStatus</code>	Top-level bundle export outcome.
class	<code>DriverAuditEventType</code>	Read-only event types used by the Driver Studio audit timeline.
class	<code>DriverAuditEvent</code>	One immutable audit event for chain-of-custody displays.
class	<code>DriverAuditTrail</code>	Read-only audit trail plus deterministic trail hash.
class	<code>DriverEvidenceRecord</code>	Studio-ready evidence summary for one reviewed driver.
class	<code>EvidenceBundleManifest</code>	Deterministic manifest describing an exported evidence bundle.
class	<code>DriverEvidenceBundle</code>	Frozen read-only packet for audit export and Driver Studio panels.
method	<code>DriverEvidenceBundle.to_dict(self) -> Mapping[str, Any]</code>	Return a canonical JSON-compatible mapping.
method	<code>DriverEvidenceBundle.to_json(self) -> str</code>	Return a deterministic JSON export string.
class	<code>EvidenceBundleExporter</code>	Create and verify read-only driver evidence bundles.
method	<code>EvidenceBundleExporter.capability_matrix(self) -> Mapping[str, bool]</code>	Return the export-layer authority boundary for GUI display.
method	<code>EvidenceBundleExporter.export_batch_review(self, batch_report: DriverBatchReviewReport, *, regression_reports: Sequence[DriverRegressionReport] Mapping[str, DriverRegressionReport] None=None, created_by: str='tds-admin', created_at: str='undated', actor_role: str='admin', policy_snapshot: Mapping[str, Any] None=None, public_signature_metadata: Mapping[str, Mapping[str, Any]] None=None, bundle_id: str None=None) -> DriverEvidenceBundle</code>	Freeze one batch review report into a deterministic evidence packet.
method	<code>EvidenceBundleExporter.verify_bundle(self, bundle: DriverEvidenceBundle Mapping[str, Any] str) -> EvidenceIntegrityStatus</code>	Verify a bundle by recomputing its deterministic bundle hash.

staqtapp_tds.drivers.registry

Type	Call / Class	Summary
class	<code>DriverState</code>	
class	<code>RegistryError</code>	
class	<code>DriverRecord</code>	value model fields: manifest, state, signature, test_report_hash
class	<code>DriverRegistry</code>	Minimal in-memory registry contract used for foundation tests.
method	<code>DriverRegistry.add_candidate(self, manifest: DriverManifest, *, test_report_hash: str None=None) -> DriverRecord</code>	
method	<code>DriverRegistry.approve(self, driver_id: str) -> DriverRecord</code>	

Type	Call / Class	Summary
method	<code>DriverRegistry.attach_signature(self, driver_id: str, signature: str) -> DriverRecord</code>	
method	<code>DriverRegistry.activate(self, driver_id: str) -> DriverRecord</code>	
method	<code>DriverRegistry.retire(self, driver_id: str) -> DriverRecord</code>	
method	<code>DriverRegistry.revoke(self, driver_id: str) -> DriverRecord</code>	
method	<code>DriverRegistry.require(self, driver_id: str) -> DriverRecord</code>	

staqtapp_tds.drivers.review

Type	Call / Class	Summary
function	<code>batch_review_capability_matrix(policy: BatchReviewPolicy None=None) -> Mapping[str, bool]</code>	Convenience function for displaying Admin batch review authority.
class	<code>ReviewAction</code>	Admin-requested review action for one driver report.
class	<code>ReviewDecisionStatus</code>	Per-driver batch review outcome.
class	<code>BatchReviewStatus</code>	Top-level status for an admin batch review run.
class	<code>ReviewFault</code>	Structured fault for Admin/Studio review surfaces.
class	<code>BatchReviewPolicy</code>	Authority and evidence policy for admin batch review.
class	<code>DriverReviewItem</code>	One report plus the admin's requested action and explanation.
class	<code>DriverReviewDecision</code>	Immutable per-driver decision generated from regression evidence.
method	<code>DriverReviewDecision.approved(self) -> bool</code>	
class	<code>DriverBatchReviewReport</code>	Deterministic batch report for Admin and future Driver Studio.
method	<code>DriverBatchReviewReport.approved_driver_ids(self) -> tuple[str, ...]</code>	
method	<code>DriverBatchReviewReport.held_driver_ids(self) -> tuple[str, ...]</code>	
class	<code>DriverBatchReviewBoard</code>	Review deterministic regression reports without expanding trust authority.
method	<code>DriverBatchReviewBoard.capability_matrix(self) -> Mapping[str, bool]</code>	Return the Admin review authority map for Studio display.
method	<code>DriverBatchReviewBoard.review_reports(self, reports: Sequence[DriverRegressionReport DriverReviewItem Mapping[str, Any]], *, reviewer_id: str='admin', requested_action: ReviewAction str=ReviewAction.APPROVE, rationale: str='', action_overrides: Mapping[str, ReviewAction str] None=None, registry: DriverRegistry None=None, apply_registry: bool=False, batch_id: str None=None) -> DriverBatchReviewReport</code>	Review a batch of regression reports and return deterministic decisions.

staqtapp_tds.drivers.signature

Type	Call / Class	Summary
function	<code>sign_payload(payload: bytes, *, signer: str, secret: bytes) -> str</code>	
function	<code>verify_signature(payload: bytes, signature: str, *, signer: str, secret: bytes) -> bool</code>	
class	<code>SignatureVerdict</code>	
class	<code>SignaturePolicy</code>	Small trust-policy model used by v3.0.6 tests and future Studio lessons.
method	<code>SignaturePolicy.approve_signer(self, signer: str, secret: bytes) -> None</code>	
method	<code>SignaturePolicy.revoke(self, signature: str) -> None</code>	

Type	Call / Class	Summary
method	<code>SignaturePolicy.evaluate(self, payload: bytes, signature: str None) -> SignatureVerdict</code>	

Driver Studio support surfaces

staqtapp_tds.drivers.studio

Type	Call / Class	Summary
function	<code>studio_instruction_reference() -> Mapping[str, Mapping[str, object]]</code>	Return compact instruction reference data for the future Learn panel.
function	<code>run_studio_quick_test(source: str, *, signer: str='studio-admin', secret: bytes=...) -> DriverStudioQuickTestReport</code>	Run the non-GUI Driver Studio Class A certification path.
function	<code>studio_readonly_capability_matrix() -> Mapping[str, bool]</code>	Convenience function for displaying Studio read-only authority.
class	<code>StudioGate</code>	Ordered Driver Studio certification gates.
class	<code>StudioGateResult</code>	value model fields: gate, passed, detail
class	<code>DriverStudioSession</code>	Minimal gated certification session for future Driver Studio UX tests.
method	<code>DriverStudioSession.pass_gate(self, gate: StudioGate) -> None</code>	
method	<code>DriverStudioSession.next_gate(self) -> StudioGate</code>	
class	<code>DriverStudioQuickTestReport</code>	Immutable report emitted by the Class A Studio quick test.
method	<code>DriverStudioQuickTestReport.passed_gates(self) -> tuple[str, ...]</code>	
class	<code>StudioPanelKind</code>	Stable panel identifiers for the future PyQt5 Driver Studio.
class	<code>StudioPanelStatus</code>	Read-only rendering status for one Studio panel.
class	<code>StudioConsoleStatus</code>	Top-level read-only console status derived from an evidence bundle.
class	<code>DriverStudioQueueItem</code>	One driver row for the Studio review queue panel.
method	<code>DriverStudioQueueItem.needs_attention(self) -> bool</code>	
class	<code>DriverStudioRiskCard</code>	Human-readable, read-only driver risk summary for Studio inspectors.
class	<code>DriverStudioEventRow</code>	Compact event-console row derived from the audit trail.
class	<code>DriverStudioPanelSnapshot</code>	One immutable panel payload for a PyQt5 view model to render.
class	<code>DriverStudioConsoleSnapshot</code>	Complete read-only evidence-console snapshot.
method	<code>DriverStudioConsoleSnapshot.panel(self, kind: StudioPanelKind str) -> DriverStudioPanelSnapshot</code>	
method	<code>DriverStudioConsoleSnapshot.to_dict(self) -> Mapping[str, Any]</code>	
method	<code>DriverStudioConsoleSnapshot.to_json(self) -> str</code>	
class	<code>DriverStudioReadOnlyConsole</code>	Read-only evidence-console model for the future PyQt5 Driver Studio.
method	<code>DriverStudioReadOnlyConsole.capability_matrix(self) -> Mapping[str, bool]</code>	Return the read-only Studio authority boundary.
method	<code>DriverStudioReadOnlyConsole.open_bundle(self, bundle: DriverEvidenceBundle Mapping[str, Any] str, *, selected_driver_id: str None=None) -> DriverStudioConsoleSnapshot</code>	Load an evidence bundle into read-only Studio panel snapshots.

staqtapp_tds.drivers.studio_actions

Type	Call / Class	Summary
function	<code>studio_admin_review_capability_matrix(policy: StudioReviewSubmissionPolicy None=None) -> Mapping[str, bool]</code>	Convenience function for displaying Studio admin-action authority.

Type	Call / Class	Summary
class	StudioReviewSubmissionStatus	Top-level result for Studio-submitted admin review actions.
class	StudioReviewActionStatus	Per-driver status for a Studio action request.
class	StudioReviewSubmissionPolicy	Input and routing policy for Driver Studio admin action submission.
class	StudioReviewActionRequest	One admin action captured by Studio for authority review.
class	StudioReviewActionEvent	Deterministic action-submission event for Studio audit surfaces.
class	StudioReviewActionDecision	Per-driver result after Studio records or routes an admin action.
class	StudioReviewSubmissionReport	Immutable report for Driver Studio admin action submission.
method	StudioReviewSubmissionReport.accepted_driver_ids(self) -> tuple[str, ...]	
method	StudioReviewSubmissionReport.to_dict(self) -> Mapping[str, Any]	
method	StudioReviewSubmissionReport.to_json(self) -> str	
class	DriverStudioAdminReviewActions	Bounded Studio layer for submitting admin review actions.
method	DriverStudioAdminReviewActions.capability_matrix(self) -> Mapping[str, bool]	Return the Studio admin-action authority map.
method	DriverStudioAdminReviewActions.submit_actions(self, console_or_bundle: DriverStudioConsoleSnapshot DriverEvidenceBundle Mapping[str, Any] str, actions: Sequence[StudioReviewActionRequest Mapping[str, Any]], *, regression_reports: Sequence[DriverRegressionReport] Mapping[str, DriverRegressionReport] None=None, review_board: DriverBatchReviewBoard None=None, registry: DriverRegistry None=None, request_registry_approval: bool=False, submitted_at: str='undated', submission_id: str None=None, authority_batch_id: str None=None) -> StudioReviewSubmissionReport	Validate, audit, and optionally route Studio review actions.

staqtapp_tds.drivers.studio_builder

Type	Call / Class	Summary
function	studio_manual_builder_capability_matrix() -> Mapping[str, bool]	Convenience function for displaying the manual builder boundary.
class	StudioManualProposalStatus	Stable status values for manual Studio proposal workbench actions.
class	StudioManualDriverTask	Human-entered task fields for a bounded TDDL driver proposal.
class	StudioManualProposalPreview	Deterministic preview payload for the cockpit manual builder panel.
class	StudioManualProposalReport	Manual builder report routed through Driver Foundry.
class	DriverStudioManualProposalBuilder	Manual driver-proposal workbench for the Driver Studio cockpit.
method	DriverStudioManualProposalBuilder.capability_matrix(self) -> Mapping[str, bool]	Return the workbench authority boundary.
method	DriverStudioManualProposalBuilder.build_source(self, task: StudioManualDriverTask) -> str	Render a deterministic TDDL source proposal from form fields.
method	DriverStudioManualProposalBuilder.preview_task(self, task: StudioManualDriverTask) -> StudioManualProposalPreview	Return source preview and metrics without compiling or executing.
method	DriverStudioManualProposalBuilder.propose_task(self, task: StudioManualDriverTask, *, fixtures: Mapping[str, Any] None=None) -> StudioManualProposalReport	Generate source and route it through Driver Foundry only.

Driver Studio PyQt5 cockpit surfaces

staqtapp_tds.studio_pyqt5.app

Type	Call / Class	Summary
function	<code>run_driver_studio_app(bundle: Any None=None, *, argv: Sequence[str] None=None, bridge: StudioQtBridge None=None, theme: StudioQtTheme None=None) -> int</code>	Run the optional PyQt5 cockpit shell.

staqtapp_tds.studio_pyqt5.availability

Type	Call / Class	Summary
function	<code>pyqt5_availability() -> PyQt5Availability</code>	Return whether PyQt5 can be imported without importing it eagerly.
function	<code>pyqt5_available() -> bool</code>	Boolean convenience wrapper for optional shell launchers.
function	<code>require_pyqt5() -> None</code>	Fail explicitly when a caller tries to create a GUI without PyQt5.
class	<code>PyQt5UnavailableError</code>	Raised only when a caller explicitly asks to construct the Qt shell.
class	<code>PyQt5Availability</code>	Stable report used by tests, launchers, and diagnostics.

staqtapp_tds.studio_pyqt5.bridge

Type	Call / Class	Summary
function	<code>studio_pyqt5_shell_capability_matrix() -> Mapping[str, bool]</code>	Convenience function for rendering shell authority boundaries.
class	<code>StudioQtPanelViewModel</code>	Widget-neutral payload prepared for a PyQt5 panel.
class	<code>StudioQtShellState</code>	Immutable shell state consumed by optional Qt widgets.
method	<code>StudioQtShellState.panel(self, kind: StudioPanelKind str) -> StudioQtPanelViewModel</code>	
class	<code>StudioQtBridge</code>	Headless bridge for the v3.1.13 Driver Studio PyQt5 cockpit shell.
method	<code>StudioQtBridge.capability_matrix(self) -> Mapping[str, bool]</code>	Return the GUI shell authority boundary.
method	<code>StudioQtBridge.console_snapshot(self) -> DriverStudioConsoleSnapshot None</code>	
method	<code>StudioQtBridge.load_bundle(self, bundle: DriverEvidenceBundle Mapping[str, Any] str, *, selected_driver_id: str None=None) -> StudioQtShellState</code>	Load an evidence bundle into immutable shell state.
method	<code>StudioQtBridge.select_driver(self, driver_id: str None) -> StudioQtShellState</code>	Re-render the current bundle with a selected queue row.
method	<code>StudioQtBridge.shell_state(self) -> StudioQtShellState</code>	Return immutable state even before a bundle has been opened.
method	<code>StudioQtBridge.hydrated_shell_state(self)</code>	Return GUI-ready hydrated cockpit state.
method	<code>StudioQtBridge.hydrate_panel(self, kind: StudioPanelKind str)</code>	Return one hydrated panel by kind.
method	<code>StudioQtBridge.manual_builder_form_schema(self)</code>	Return stable form fields for the Manual Driver Builder Qt panel.
method	<code>StudioQtBridge.panel_refresh_contracts(self)</code>	Return v3.1.12 refresh contracts for live cockpit panels.
method	<code>StudioQtBridge.live_event_bridge(self, *, max_events: int=256)</code>	Create a bounded live-event bridge around this Studio bridge.
method	<code>StudioQtBridge.live_panel_runtime(self, *, max_events: int=256)</code>	Create a v3.1.13 live panel runtime around this Studio bridge.
method	<code>StudioQtBridge.review_workflow_console(self, *, max_events: int=256)</code>	Create a v3.1.14 Review Workflow Console around this Studio bridge.
method	<code>StudioQtBridge.evidence_timeline(self, *, max_events: int=256)</code>	Create a v3.1.16 Evidence Timeline around this Studio bridge.
method	<code>StudioQtBridge.risk_intelligence_cards(self, *, max_events: int=256)</code>	Create v3.1.16 Risk Intelligence Cards around this Studio bridge.
method	<code>StudioQtBridge.manual_builder_ui_runtime(self, *, max_events: int=256)</code>	Create the v3.1.18 Manual Builder UI Runtime around this bridge.
method	<code>StudioQtBridge.export_audit_console(self, *, max_events: int=256)</code>	Create the v3.1.20 Export / Audit Console around this bridge.
method	<code>StudioQtBridge.export_integrity_workflow(self, *, max_events: int=256)</code>	Create the v3.1.20 Export Integrity Workflow around this bridge.
method	<code>StudioQtBridge.visual_quality_review(self)</code>	Run the v3.1.18 static PyQt5 cockpit visual-quality review.

Type	Call / Class	Summary
method	<code>StudioQtBridge.build_action_request(self, driver_id: str, action: ReviewAction str, *, reviewer_id: str='studio-admin', rationale: str='', source_panel: StudioPanelKind str=StudioPanelKind.DRIVER_QUEUE, tags: Sequence[str]=()) -> StudioReviewActionRequest</code>	Create the safe request object used by action buttons.
method	<code>StudioQtBridge.submit_review_action(self, request: StudioReviewActionRequest Mapping[str, Any], *, submitted_at: str='undated') -> StudioReviewSubmissionReport</code>	Record an action through v3.1.8; does not route authority by default.
method	<code>StudioQtBridge.preview_manual_driver_task(self, task: StudioManualDriverTask) -> StudioManualProposalPreview</code>	Render a manual-builder proposal preview without trust authority.
method	<code>StudioQtBridge.propose_manual_driver_task(self, task: StudioManualDriverTask, *, fixtures: Mapping[str, Any] None=None) -> StudioManualProposalReport</code>	Route a cockpit-built driver task through Driver Foundry only.

staqtapp_tds.studio_pyqt5.evidence_timeline

Type	Call / Class	Summary
function	<code>studio_evidence_timeline_capability_matrix() -> Mapping[str, bool]</code>	Convenience helper for displaying v3.1.16 timeline boundaries.
class	<code>StudioDriverLifecycleStage</code>	Lifecycle stages rendered by the v3.1.16 Evidence Timeline.
class	<code>StudioEvidenceTimelineItem</code>	One chronological lifecycle item surfaced by Driver Studio.
method	<code>StudioEvidenceTimelineItem.registry_related(self) -> bool</code>	
method	<code>StudioEvidenceTimelineItem.as_row(self) -> Mapping[str, Any]</code>	
class	<code>StudioRegistryObservationItem</code>	Registry-related timeline item extracted for audit/export review.
method	<code>StudioRegistryObservationItem.as_row(self) -> Mapping[str, Any]</code>	
class	<code>StudioEvidenceTimelineIntegrityCard</code>	Integrity/export readiness summary for the Evidence Timeline.
method	<code>StudioEvidenceTimelineIntegrityCard.as_card(self) -> Mapping[str, Any]</code>	
class	<code>StudioEvidenceTimelineState</code>	Complete v3.1.16 Evidence Timeline view-model.
method	<code>StudioEvidenceTimelineState.latest_stage(self) -> StudioDriverLifecycleStage None</code>	
method	<code>StudioEvidenceTimelineState.export_ready(self) -> bool</code>	
method	<code>StudioEvidenceTimelineState.registry_observation_count(self) -> int</code>	
method	<code>StudioEvidenceTimelineState.items_for_stage(self, stage: StudioDriverLifecycleStage str) -> tuple[StudioEvidenceTimelineItem, ...]</code>	
method	<code>StudioEvidenceTimelineState.signal_payload(self) -> Mapping[str, Any]</code>	
class	<code>StudioEvidenceTimeline</code>	Chronological trust-history layer for Studio & Evidence.
method	<code>StudioEvidenceTimeline.capability_matrix(self) -> Mapping[str, bool]</code>	
method	<code>StudioEvidenceTimeline.current_state(self, runtime_state: StudioPanelRuntimeState None=None) -> StudioEvidenceTimelineState</code>	
method	<code>StudioEvidenceTimeline.signal_payload(self) -> Mapping[str, Any]</code>	

staqtapp_tds.studio_pyqt5.export_audit

Type	Call / Class	Summary
function	<code>studio_export_audit_capability_matrix() -> Mapping[str, bool]</code>	Convenience helper for displaying v3.1.20 export/audit boundaries.
class	<code>StudioExportAuditStatus</code>	Top-level Export / Audit Console readiness status.
class	<code>StudioExportAuditIntegrityItem</code>	One manifest/checklist integrity item for a future export packet.

Type	Call / Class	Summary
method	StudioExportAuditIntegrityItem.ready(self) -> bool	
method	StudioExportAuditIntegrityItem.as_row(self) -> Mapping[str, Any]	
class	StudioExportAuditChecklist	Stable checklist rendered beside the packet preview.
method	StudioExportAuditChecklist.required_count(self) -> int	
method	StudioExportAuditChecklist.ready_count(self) -> int	
method	StudioExportAuditChecklist.missing_items(self) -> tuple[str, ...]	
method	StudioExportAuditChecklist.ready(self) -> bool	
method	StudioExportAuditChecklist.as_card(self) -> Mapping[str, Any]	
class	StudioExportAuditReadinessCard	Compact readiness summary for the selected export/audit packet.
method	StudioExportAuditReadinessCard.as_card(self) -> Mapping[str, Any]	
class	StudioExportAuditManifest	Deterministic manifest preview for a future export/audit packet.
method	StudioExportAuditManifest.with_manifest_hash(self) -> 'StudioExportAuditManifest'	
method	StudioExportAuditManifest.as_manifest(self, *, include_hash: bool=True) -> Mapping[str, Any]	
class	StudioExportAuditPacketPreview	Reviewable packet preview assembled from visible Studio evidence.
method	StudioExportAuditPacketPreview.as_payload(self) -> Mapping[str, Any]	
class	StudioExportAuditConsoleState	Complete v3.1.20 Export / Audit Console view-model.
method	StudioExportAuditConsoleState.signal_payload(self) -> Mapping[str, Any]	
class	StudioExportAuditConsole	Prepare selected-driver export/audit packet previews.
method	StudioExportAuditConsole.capability_matrix(self) -> Mapping[str, bool]	
method	StudioExportAuditConsole.current_state(self, runtime_state: StudioPanelRuntimeState None=None, *, performance_report: Any Mapping[str, Any] None=None) -> StudioExportAuditConsoleState	
method	StudioExportAuditConsole.packet_preview(self, runtime_state: StudioPanelRuntimeState None=None, *, performance_report: Any Mapping[str, Any] None=None) -> StudioExportAuditPacketPreview	
method	StudioExportAuditConsole.signal_payload(self) -> Mapping[str, Any]	

staqtapp_tds.studio_pyqt5.export_integrity_workflow

Type	Call / Class	Summary
function	studio_export_integrity_workflow_capability_matrix() -> Mapping[str, bool]	Convenience helper for displaying v3.1.20 export-integrity boundaries.
class	StudioExportIntegrityWorkflowStatus	Top-level v3.1.20 export-integrity workflow status.
class	StudioExportIntegrityCheckpointStatus	Status for one review-safe export-integrity checkpoint.
class	StudioExportIntegrityCheckpoint	One progressive checkpoint in the export-integrity workflow.
method	StudioExportIntegrityCheckpoint.ok(self) -> bool	
method	StudioExportIntegrityCheckpoint.as_row(self) -> Mapping[str, Any]	
class	StudioExportIntegrityManifestComparison	Comparison between the current manifest and an expected manifest/hash.

Type	Call / Class	Summary
method	StudioExportIntegrityManifestComparison.as_payload(self) -> Mapping[str, Any]	
class	StudioExportIntegrityReviewGate	Review-safe readiness gate for handoff to export/review tooling.
method	StudioExportIntegrityReviewGate.as_card(self) -> Mapping[str, Any]	
class	StudioExportIntegrityWorkflowState	Complete v3.1.20 Export Integrity Workflow state.
method	StudioExportIntegrityWorkflowState.blocking_checkpoint_ids(self) -> tuple[str, ...]	
method	StudioExportIntegrityWorkflowState.signal_payload(self) -> Mapping[str, Any]	
class	StudioExportIntegrityWorkflow	Verify export/audit packet previews for review-safe handoff.
method	StudioExportIntegrityWorkflow.capability_matrix(self) -> Mapping[str, bool]	
method	StudioExportIntegrityWorkflow.current_state(self, runtime_state: StudioPanelRuntimeState None=None, *, performance_report: Any Mapping[str, Any] None=None, expected_manifest: Mapping[str, Any] None=None, expected_manifest_hash: str None=None, expected_packet_hash: str None=None) -> StudioExportIntegrityWorkflowState	
method	StudioExportIntegrityWorkflow.signal_payload(self) -> Mapping[str, Any]	

staqtapp_tds.studio_pyqt5.hydration

Type	Call / Class	Summary
function	manual_builder_form_schema() -> tuple[StudioFormField, ...]	Return stable form fields for the Manual Driver Builder panel.
function	studio_cockpit_hydration_capability_matrix() -> Mapping[str, bool]	Convenience helper for displaying hydration boundaries.
class	StudioTableColumn	Stable table-column metadata for a hydrated Studio panel.
class	StudioPanelCard	Card payload for evidence, risk, registry, integrity, and builder views.
class	StudioTimelineItem	Event stream row prepared for a timeline/list widget.
class	StudioPanelActionDescriptor	Button/action metadata for review-intent controls.
class	StudioFormField	Manual-builder field metadata for a real Qt form.
class	StudioHydratedPanel	A GUI-ready panel model with richer surface semantics.
class	StudioHydratedCockpitState	Whole-cockpit hydration result consumed by a PyQt5 main window.
method	StudioHydratedCockpitState.panel(self, kind: StudioPanelKind str) -> StudioHydratedPanel	
class	StudioCockpitHydrator	Pure-Python hydrator for optional Driver Studio PyQt5 views.
method	StudioCockpitHydrator.capability_matrix(self) -> Mapping[str, bool]	Return the hydration authority boundary.
method	StudioCockpitHydrator.hydrate(self, state: StudioQtShellState) -> StudioHydratedCockpitState	Hydrate an immutable shell state for concrete Qt widgets.
method	StudioCockpitHydrator.hydrate_panel(self, panel: StudioQtPanelViewModel, *, state: StudioQtShellState) -> StudioHydratedPanel	Hydrate one compact panel view-model.

staqtapp_tds.studio_pyqt5.icons

Type	Call / Class	Summary
function	studio_svg_icon(name: str) -> str	Return a named Studio SVG icon or raise KeyError for unknown names.

staqtapp_tds.studio_pyqt5.live

Type	Call / Class	Summary
function	<code>studio_panel_refresh_contracts() -> tuple[StudioPanelRefreshContract, ...]</code>	Return stable panel refresh contracts for the live cockpit.
function	<code>studio_live_event_bridge_capability_matrix() -> Mapping[str, bool]</code>	Convenience helper for displaying v3.1.12 live bridge boundaries.
class	<code>StudioLiveEventKind</code>	Stable event kinds emitted by the live cockpit bridge.
class	<code>StudioLiveEvent</code>	One bounded event-console item for the PyQt5 live bridge.
method	<code>StudioLiveEvent.as_row(self) -> Mapping[str, Any]</code>	Return a compact row suitable for Qt table/list models.
class	<code>StudioCockpitSelection</code>	Selected bundle/driver identity coordinated across Studio panels.
class	<code>StudioPanelRefreshContract</code>	Refresh policy metadata for one cockpit panel.
class	<code>StudioLiveCockpitState</code>	Complete live-state payload consumed by a PyQt5 shell.
method	<code>StudioLiveCockpitState.event_retention_gap(self) -> bool</code>	Whether the retained stream has dropped earlier live events.
method	<code>StudioLiveCockpitState.latest_event_id(self) -> str None</code>	
method	<code>StudioLiveCockpitState.panel(self, kind: StudioPanelKind str) -> StudioHydratedPanel</code>	
method	<code>StudioLiveCockpitState.events_since(self, cursor: int) -> tuple[StudioLiveEvent, ...]</code>	Return retained events with sequence numbers above ``cursor``.
method	<code>StudioLiveCockpitState.has_retention_gap_since(self, cursor: int) -> bool</code>	Return True when older events were dropped before ``cursor`` caught up.
method	<code>StudioLiveCockpitState.signal_payload(self) -> Mapping[str, Any]</code>	Return JSON-friendly payload data for Qt signal emission.
class	<code>StudioCockpitEventBridge</code>	Bounded live-update bridge for the Driver Studio cockpit.
method	<code>StudioCockpitEventBridge.generation(self) -> int</code>	
method	<code>StudioCockpitEventBridge.cursor(self) -> int</code>	
method	<code>StudioCockpitEventBridge.events(self) -> tuple[StudioLiveEvent, ...]</code>	
method	<code>StudioCockpitEventBridge.dropped_event_count(self) -> int</code>	
method	<code>StudioCockpitEventBridge.retained_cursor_floor(self) -> int</code>	
method	<code>StudioCockpitEventBridge.capability_matrix(self) -> Mapping[str, bool]</code>	Return live-bridge capabilities and denied authority.
method	<code>StudioCockpitEventBridge.current_state(self) -> StudioLiveCockpitState</code>	Return current hydrated live state without recording a new event.
method	<code>StudioCockpitEventBridge.refresh(self, *, reason: str='manual refresh', timestamp: str='undated') -> StudioLiveCockpitState</code>	Emit a snapshot-refresh event and return the hydrated state.
method	<code>StudioCockpitEventBridge.load_bundle(self, bundle: DriverEvidenceBundle Mapping[str, Any] str, *, selected_driver_id: str None=None, timestamp: str='undated') -> StudioLiveCockpitState</code>	Load a bundle through StudioQtBridge and record a safe UI event.
method	<code>StudioCockpitEventBridge.select_driver(self, driver_id: str None, *, timestamp: str='undated') -> StudioLiveCockpitState</code>	Re-render selection through the bridge and record selected context.
method	<code>StudioCockpitEventBridge.refresh_panel(self, kind: StudioPanelKind str, *, timestamp: str='undated') -> StudioHydratedPanel</code>	Hydrate one panel and record a panel-refresh event.
method	<code>StudioCockpitEventBridge.submit_review_action(self, request: StudioReviewActionRequest Mapping[str, Any], *, submitted_at: str='undated') -> StudioReviewSubmissionReport</code>	Submit review intent through the existing action layer and emit an event.
method	<code>StudioCockpitEventBridge.preview_manual_driver_task(self, task: StudioManualDriverTask, *, timestamp: str='undated') -> StudioManualProposalPreview</code>	Preview a manual task and emit a proposal-preview event.
method	<code>StudioCockpitEventBridge.propose_manual_driver_task(self, task: StudioManualDriverTask, *, fixtures: Mapping[str, Any] None=None, timestamp: str='undated') -> StudioManualProposalReport</code>	Route manual task through Foundry and emit a proposal event.

Type	Call / Class	Summary
method	<code>StudioCockpitEventBridge.events_since(self, cursor: int) -> tuple[StudioLiveEvent, ...]</code>	Return retained live events after cursor.
method	<code>StudioCockpitEventBridge.signal_payload(self) -> Mapping[str, Any]</code>	Return current state as a compact Qt-signal-friendly mapping.

staqtapp_tds.studio_pyqt5.main_window

Type	Call / Class	Summary
function	<code>create_driver_studio_window(*, bridge: StudioQtBridge None=None, theme: StudioQtTheme None=None) -> DriverStudioMainWindow</code>	Create the real Qt main window when PyQt5 is installed.

staqtapp_tds.studio_pyqt5.manual_builder_runtime

Type	Call / Class	Summary
function	<code>studio_manual_builder_ui_runtime_capability_matrix() -> Mapping[str, bool]</code>	Return the v3.1.18 UI runtime capability and boundary matrix.
function	<code>studio_qt_visual_quality_capability_matrix() -> Mapping[str, bool]</code>	Return the static visual-review boundary.
function	<code>studio_qt_visual_quality_review(*, form_fields: Sequence[StudioFormField] None=None, source: str='', theme: StudioQtTheme None=None, joins: Sequence[StudioManualBuilderJoin]=()) -> StudioQtVisualQualityReport</code>	Run a static, headless review of cockpit readability and flow.
class	<code>StudioManualBuilderRuntimeStatus</code>	Stable state machine values for the manual builder UI runtime.
class	<code>StudioManualBuilderRuntimeStep</code>	Ordered UI steps used by the joined cockpit flow.
class	<code>StudioManualBuilderJoin</code>	One join between the builder and a neighboring Studio panel.
method	<code>StudioManualBuilderJoin.signal_payload(self) -> Mapping[str, Any]</code>	
class	<code>StudioQtVisualQualityRule</code>	One static cockpit visual-quality rule for CI and headless review.
class	<code>StudioQtVisualQualityReport</code>	Static visual-quality review for the Driver Studio PyQt5 app.
method	<code>StudioQtVisualQualityReport.signal_payload(self) -> Mapping[str, Any]</code>	
class	<code>StudioManualBuilderRuntimeState</code>	Complete signal-friendly state for a Manual Builder UI panel.
method	<code>StudioManualBuilderRuntimeState.signal_payload(self) -> Mapping[str, Any]</code>	
class	<code>StudioManualBuilderUIRuntime</code>	Opt-in runtime surface for the Manual Driver Builder panel.
method	<code>StudioManualBuilderUIRuntime.capability_matrix(self) -> Mapping[str, bool]</code>	Return the manual-builder UI runtime boundary.
method	<code>StudioManualBuilderUIRuntime.current_state(self) -> StudioManualBuilderRuntimeState</code>	Return the most recent UI runtime state.
method	<code>StudioManualBuilderUIRuntime.default_form_payload(self) -> Mapping[str, Any]</code>	Return default form payload values matching the hydrated schema.
method	<code>StudioManualBuilderUIRuntime.task_from_form_payload(self, payload: Mapping[str, Any]) -> StudioManualDriverTask</code>	Normalize a Qt form payload into a StudioManualDriverTask.
method	<code>StudioManualBuilderUIRuntime.preview_form_payload(self, payload: Mapping[str, Any] None=None) -> StudioManualBuilderRuntimeState</code>	Preview form input as deterministic TDDL without Foundry routing.
method	<code>StudioManualBuilderUIRuntime.propose_form_payload(self, payload: Mapping[str, Any] None=None, *, fixtures: Mapping[str, Any] None=None) -> StudioManualBuilderRuntimeState</code>	Route a form payload through Driver Foundry only.
method	<code>StudioManualBuilderUIRuntime.form_fields(self) -> tuple[StudioFormField, ...]</code>	
method	<code>StudioManualBuilderUIRuntime.form_schema(self) -> tuple[StudioFormField, ...]</code>	Compatibility alias for GUI code that expects schema wording.
method	<code>StudioManualBuilderUIRuntime.quality_review(self) -> StudioQtVisualQualityReport</code>	Return the current static visual quality review.
method	<code>StudioManualBuilderUIRuntime.signal_payload(self) -> Mapping[str, Any]</code>	

staqtapp_tds.studio_pyqt5.operational_stress

Type	Call / Class	Summary
function	<code>studio_operational_stress_capability_matrix()</code> -> Mapping[str, bool]	Convenience helper for the v3.1.23 operational stress boundary.
class	<code>StudioOperationalStressStatus</code>	Top-level status for a headless operational stress run.
class	<code>StudioOperationalStressScenario</code>	Named operational stress scenarios for v3.1.23 scenario-matrix runs.
class	<code>StudioOperationalStressObservation</code>	One stress observation row for UI, logs, or release checks.
method	<code>StudioOperationalStressObservation.as_row(self)</code> -> Mapping[str, Any]	
class	<code>StudioOperationalStressReport</code>	Complete report produced by the v3.1.23 operational stress harness.
method	<code>StudioOperationalStressReport.signal_payload(self)</code> -> Mapping[str, Any]	Return JSON-friendly data for Qt/browser display or CI artifacts.
class	<code>StudioOperationalStressScenarioResult</code>	Result row for one named v3.1.23 operational stress scenario.
method	<code>StudioOperationalStressScenarioResult.name(self)</code> -> str	
method	<code>StudioOperationalStressScenarioResult.signal_payload(self)</code> -> Mapping[str, Any]	Return JSON-friendly scenario-result data.
class	<code>StudioOperationalStressScenarioMatrix</code>	Deterministic scenario matrix built on top of the operational stress harness.
method	<code>StudioOperationalStressScenarioMatrix.signal_payload(self)</code> -> Mapping[str, Any]	Return JSON-friendly matrix data for Studio, Browser, CI, or AI stress tooling.
class	<code>StudioOperationalStressHarness</code>	Headless stress runner for simultaneous Browser and Studio observers.
method	<code>StudioOperationalStressHarness.capability_matrix(self)</code> -> Mapping[str, bool]	Return stress-harness capabilities and denied authority.
method	<code>StudioOperationalStressHarness.run_scenario(self, scenario: StudioOperationalStressScenario str, *, iterations: int=32)</code> -> StudioOperationalStressScenarioResult	Run one named operational stress scenario without halting host operation.
method	<code>StudioOperationalStressHarness.run_scenario_matrix(self, scenarios: tuple[StudioOperationalStressScenario str, ...] list[StudioOperationalStressScenario str] None=None, *, iterations: int=32)</code> -> StudioOperationalStressScenarioMatrix	Run the default or supplied v3.1.23 stress scenario matrix.
method	<code>StudioOperationalStressHarness.run(self, *, iterations: int=32, browser_polls: int None=None, studio_refreshes: int None=None, include_tds_persistence: bool=True)</code> -> StudioOperationalStressReport	Run the bounded observer stress scenario and return a report.

staqtapp_tds.studio_pyqt5.panels.descriptors

Type	Call / Class	Summary
function	<code>panel_descriptor(kind: StudioPanelKind str)</code> -> StudioPanelDescriptor	
class	<code>StudioPanelDescriptor</code>	Stable panel metadata for Qt docks, tests, and documentation.

staqtapp_tds.studio_pyqt5.review_workflow

Type	Call / Class	Summary
function	<code>studio_review_rationale_templates()</code> -> tuple[StudioReviewRationaleTemplate, ...]	Return stable rationale templates for the Review Workflow Console.
function	<code>studio_review_workflow_capability_matrix()</code> -> Mapping[str, bool]	Convenience helper for displaying v3.1.14 workflow boundaries.
class	<code>StudioReviewWorkflowStatus</code>	Top-level readiness state for the review workflow console.
class	<code>StudioReviewRationaleTemplate</code>	Stable rationale template surfaced by the Review Workflow Console.
method	<code>StudioReviewRationaleTemplate.render(self, *, driver_id: str None=None, reason: str None=None)</code> -> str	Render a bounded human-editable rationale string.

Type	Call / Class	Summary
method	StudioReviewRationaleTemplate.as_row(self) -> Mapping[str, Any]	
class	StudioReviewActionEligibility	Qt-ready action metadata with explicit eligibility explanation.
method	StudioReviewActionEligibility.as_button(self) -> Mapping[str, Any]	
class	StudioReviewWorkflowItem	One driver row in the Review Workflow Console.
method	StudioReviewWorkflowItem.approval_ready(self) -> bool	
method	StudioReviewWorkflowItem.needs_attention(self) -> bool	
method	StudioReviewWorkflowItem.action(self, requested_action: ReviewAction str) -> StudioReviewActionEligibility	
method	StudioReviewWorkflowItem.as_row(self) -> Mapping[str, Any]	
class	StudioReviewHistoryEntry	Review-relevant history item derived from audit/live event surfaces.
method	StudioReviewHistoryEntry.as_row(self) -> Mapping[str, Any]	
class	StudioReviewWorkflowConsoleState	Complete Review Workflow Console view-model.
method	StudioReviewWorkflowConsoleState.attention_count(self) -> int	
method	StudioReviewWorkflowConsoleState.ready_count(self) -> int	
method	StudioReviewWorkflowConsoleState.item(self, driver_id: str) -> StudioReviewWorkflowItem	
method	StudioReviewWorkflowConsoleState.template(self, template_id: str) -> StudioReviewRationaleTemplate	
method	StudioReviewWorkflowConsoleState.action_for_selected(self, requested_action: ReviewAction str) -> StudioReviewActionEligibility	
method	StudioReviewWorkflowConsoleState.signal_payload(self) -> Mapping[str, Any]	
class	StudioReviewWorkflowConsole	Decision-support layer for Studio review workflows.
method	StudioReviewWorkflowConsole.capability_matrix(self) -> Mapping[str, bool]	
method	StudioReviewWorkflowConsole.current_state(self, runtime_state: StudioPanelRuntimeState None=None) -> StudioReviewWorkflowConsoleState	
method	StudioReviewWorkflowConsole.build_request(self, driver_id: str, action: ReviewAction str, *, reviewer_id: str='studio-admin', rationale: str='', source_panel: StudioPanelKind str=StudioPanelKind.RISK_CARD, tags: Sequence[str]=()) -> StudioReviewActionRequest	
method	StudioReviewWorkflowConsole.build_selected_request(self, action: ReviewAction str, *, reviewer_id: str='studio-admin', rationale: str='', template_id: str None=None, tags: Sequence[str]=()) -> StudioReviewActionRequest	
method	StudioReviewWorkflowConsole.submit_selected_action(self, action: ReviewAction str, *, reviewer_id: str='studio-admin', rationale: str='', template_id: str None=None, submitted_at: str='undated', tags: Sequence[str]=()) -> tuple[StudioReviewSubmissionReport, StudioPanelRuntimeState, StudioReviewWorkflowConsoleState]	
method	StudioReviewWorkflowConsole.signal_payload(self) -> Mapping[str, Any]	

staqtapp_tds.studio_pyqt5.risk_intelligence

Type	Call / Class	Summary
function	<code>studio_risk_intelligence_capability_matrix()</code> -> Mapping[str, bool]	Convenience helper for displaying v3.1.16 Risk Intelligence boundaries.
class	<code>StudioRiskIntelligenceBand</code>	Stable risk-pressure bands used by the Driver Studio cockpit.
class	<code>StudioRiskIntelligenceFactor</code>	One explainable risk factor derived from evidence, not authority.
method	<code>StudioRiskIntelligenceFactor.as_row(self)</code> -> Mapping[str, Any]	
class	<code>StudioRiskIntelligenceCard</code>	Admin-facing risk card with timeline and review context.
method	<code>StudioRiskIntelligenceCard.attention_required(self)</code> -> bool	
method	<code>StudioRiskIntelligenceCard.as_card(self)</code> -> Mapping[str, Any]	
method	<code>StudioRiskIntelligenceCard.signal_payload(self)</code> -> Mapping[str, Any]	
class	<code>StudioRiskIntelligenceState</code>	Complete v3.1.16 Risk Intelligence view-model.
method	<code>StudioRiskIntelligenceState.card_for_driver(self, driver_id: str)</code> -> StudioRiskIntelligenceCard	
method	<code>StudioRiskIntelligenceState.signal_payload(self)</code> -> Mapping[str, Any]	
class	<code>StudioRiskIntelligenceCards</code>	Evidence-facing risk analysis layer for Driver Studio.
method	<code>StudioRiskIntelligenceCards.capability_matrix(self)</code> -> Mapping[str, bool]	
method	<code>StudioRiskIntelligenceCards.current_state(self, runtime_state: StudioPanelRuntimeState None=None)</code> -> StudioRiskIntelligenceState	
method	<code>StudioRiskIntelligenceCards.signal_payload(self)</code> -> Mapping[str, Any]	

staqtapp_tds.studio_pyqt5.runtime

Type	Call / Class	Summary
function	<code>studio_live_panel_runtime_capability_matrix()</code> -> Mapping[str, bool]	Convenience helper for displaying v3.1.13 runtime boundaries.
class	<code>StudioPanelDirtyMark</code>	One reason that a cockpit panel should refresh.
method	<code>StudioPanelDirtyMark.as_row(self)</code> -> Mapping[str, Any]	
class	<code>StudioPanelRefreshPacket</code>	Qt-ready immutable panel refresh payload.
method	<code>StudioPanelRefreshPacket.kind(self)</code> -> StudioPanelKind	
method	<code>StudioPanelRefreshPacket.signal_payload(self)</code> -> Mapping[str, Any]	Return compact JSON-friendly data for Qt signals/model updates.
class	<code>StudioPanelRuntimeState</code>	Complete state produced by the live panel runtime.
method	<code>StudioPanelRuntimeState.dirty_panel_kinds(self)</code> -> tuple[StudioPanelKind, ...]	
method	<code>StudioPanelRuntimeState.packet(self, kind: StudioPanelKind str)</code> -> StudioPanelRefreshPacket	
method	<code>StudioPanelRuntimeState.signal_payload(self)</code> -> Mapping[str, Any]	Return compact state for one Qt signal emission.
class	<code>StudioLivePanelRuntime</code>	Coordinate live events into refreshed hydrated cockpit panels.
method	<code>StudioLivePanelRuntime.cursor(self)</code> -> int	
method	<code>StudioLivePanelRuntime.consumed_cursor(self)</code> -> int	
method	<code>StudioLivePanelRuntime.capability_matrix(self)</code> -> Mapping[str, bool]	
method	<code>StudioLivePanelRuntime.current_state(self, *, include_packets: bool=False)</code> -> StudioPanelRuntimeState	Return current runtime state without advancing the consumed cursor.
method	<code>StudioLivePanelRuntime.consume(self, *, include_packets: bool=True)</code> -> StudioPanelRuntimeState	Consume newly retained events and advance the runtime cursor.

Type	Call / Class	Summary
method	<code>StudioLivePanelRuntime.mark_all_dirty(self, *, reason: str='initial runtime hydration') -> StudioPanelRuntimeState</code>	Emit a snapshot refresh and return refresh packets for every panel.
method	<code>StudioLivePanelRuntime.plan_refresh(self, events: Sequence[StudioLiveEvent]) -> tuple[StudioPanelDirtyMark, ...]</code>	Map live events to deterministic panel dirty marks.
method	<code>StudioLivePanelRuntime.refresh_packets(self, dirty_marks: Sequence[StudioPanelDirtyMark], *, live_state: StudioLiveCockpitState None=None) -> tuple[StudioPanelRefreshPacket, ...]</code>	Hydrate each dirty panel once and preserve its dirty reasons.
method	<code>StudioLivePanelRuntime.load_bundle(self, bundle: DriverEvidenceBundle Mapping[str, Any] str, *, selected_driver_id: str None=None, timestamp: str='undated') -> StudioPanelRuntimeState</code>	
method	<code>StudioLivePanelRuntime.select_driver(self, driver_id: str None, *, timestamp: str='undated') -> StudioPanelRuntimeState</code>	
method	<code>StudioLivePanelRuntime.refresh(self, *, reason: str='manual refresh', timestamp: str='undated') -> StudioPanelRuntimeState</code>	
method	<code>StudioLivePanelRuntime.refresh_panel(self, kind: StudioPanelKind str, *, timestamp: str='undated') -> StudioPanelRuntimeState</code>	
method	<code>StudioLivePanelRuntime.submit_review_action(self, request: StudioReviewActionRequest Mapping[str, Any], *, submitted_at: str='undated') -> tuple[StudioReviewSubmissionReport, StudioPanelRuntimeState]</code>	
method	<code>StudioLivePanelRuntime.preview_manual_driver_task(self, task: StudioManualDriverTask, *, timestamp: str='undated') -> tuple[StudioManualProposalPreview, StudioPanelRuntimeState]</code>	
method	<code>StudioLivePanelRuntime.propose_manual_driver_task(self, task: StudioManualDriverTask, *, fixtures: Mapping[str, Any] None=None, timestamp: str='undated') -> tuple[StudioManualProposalReport, StudioPanelRuntimeState]</code>	
method	<code>StudioLivePanelRuntime.risk_intelligence_cards(self)</code>	Create v3.1.16 Risk Intelligence Cards on this live runtime.
method	<code>StudioLivePanelRuntime.visual_quality_review(self)</code>	Run the v3.1.18 static PyQt5 cockpit visual-quality review.
method	<code>StudioLivePanelRuntime.signal_payload(self) -> Mapping[str, Any]</code>	Return the current unconsumed runtime state as a Qt-friendly payload.
method	<code>StudioLivePanelRuntime.manual_builder_ui_runtime(self)</code>	Create the v3.1.18 Manual Builder UI Runtime on this live runtime.
method	<code>StudioLivePanelRuntime.review_workflow_console(self)</code>	Create a v3.1.14 Review Workflow Console on this live runtime.
method	<code>StudioLivePanelRuntime.evidence_timeline(self)</code>	Create a v3.1.16 Evidence Timeline on this live runtime.
method	<code>StudioLivePanelRuntime.export_audit_console(self)</code>	Create a v3.1.20 Export / Audit Console on this live runtime.
method	<code>StudioLivePanelRuntime.export_integrity_workflow(self)</code>	Create a v3.1.20 Export Integrity Workflow on this live runtime.

staqtapp_tds.studio_pyqt5.theme

Type	Call / Class	Summary
class	<code>StudioQtTheme</code>	Small immutable theme object for Qt widgets and tests.
method	<code>StudioQtTheme.palette(self) -> Mapping[str, str]</code>	
method	<code>StudioQtTheme.stylesheet(self) -> str</code>	Return a compact Qt stylesheet for the shell cockpit.

Metadata, provenance, spiral/trace surfaces

staqtapp_tds.metadata.aggregation

Type	Call / Class	Summary
class	TraceSetManifest	value model fields: run_id, set_id, trace_ids, created_at, set_role, rank_policy, metadata
method	TraceSetManifest.to_dict(self) -> dict[str, Any]	
method	TraceSetManifest.from_mapping(cls, data: dict[str, Any]) -> 'TraceSetManifest'	
class	AggregationRecord	value model fields: run_id, aggregation_id, output_entry, derived_from, created_at, aggregation_step, rank_score, rank_source, rank_method, rank_confidence, rank_config_id, verifier_id
method	AggregationRecord.to_dict(self) -> dict[str, Any]	
method	AggregationRecord.from_mapping(cls, data: dict[str, Any]) -> 'AggregationRecord'	

staqtapp_tds.metadata.chunk

Type	Call / Class	Summary
class	ChunkDescriptor	value model fields: chunk_id, entry_id, offset, length, stored_length, compressed, checksum, codec, runtime_config
method	ChunkDescriptor.to_dict(self) -> dict[str, Any]	

staqtapp_tds.metadata.entry

Type	Call / Class	Summary
class	EntryDescriptor	value model fields: entry_id, key, handle, hash_value, chunk_id, size, flags, runtime_config
method	EntryDescriptor.to_dict(self) -> dict[str, Any]	

staqtapp_tds.metadata.execution

Type	Call / Class	Summary
class	ProbeStatistics	value model fields: avg_probe, max_probe, load_factor, tombstones, resize_count
method	ProbeStatistics.to_dict(self) -> dict[str, Any]	
class	ExecutionCounters	value model fields: native_backend_ops, python_backend_ops, gil_released_ops, python_native_transitions, native_batch_ops, pool_reuse_count, allocator_calls
method	ExecutionCounters.native_percent(self) -> float	
method	ExecutionCounters.to_dict(self) -> dict[str, Any]	

staqtapp_tds.metadata.namespace

Type	Call / Class	Summary
class	NamespaceDescriptor	value model fields: name, path, parent, depth, entry_count, child_count, route_id
method	NamespaceDescriptor.to_dict(self) -> dict[str, Any]	

staqtapp_tds.metadata.provenance

Type	Call / Class	Summary
class	ProvenanceRecord	value model fields: object_id, parents, operation, creator, created_at, runtime_config, metadata
method	ProvenanceRecord.to_dict(self) -> dict[str, Any]	

staqtapp_tds.metadata.snapshot

Type	Call / Class	Summary
class	RuntimeSnapshot	value model fields: schema_version, created_at, uptime_seconds, performance, storage, indexes, behavior
method	RuntimeSnapshot.to_dict(self) -> dict[str, Any]	

staqtapp_tds.metadata.trace

Type	Call / Class	Summary
class	TraceRecord	value model fields: run_id, trace_id, role, created_at, content_entry, derived_from, set_ids, rank_score, rank_source, rank_method, rank_confidence, rank_config_id
method	TraceRecord.to_dict(self) -> dict[str, Any]	
method	TraceRecord.from_mapping(cls, data: dict[str, Any]) -> 'TraceRecord'	

staqtapp_tds.spiral.rank

Type	Call / Class	Summary
function	rank_traces(trace_ids: Iterable[str], scores: Iterable[float], **kwargs) -> list[SpiralRankRecord]	
function	rank_trace_run(trace_ids: Iterable[str], scores: Iterable[float], **kwargs) -> SpiralRankRun	
function	rank_trace_result(trace_ids: Iterable[str], scores: Iterable[float], **kwargs) -> TDSResult	
class	SpiralRankConfig	value model fields: score_weight, confidence_weight, depth_penalty, age_penalty, config_id
class	SpiralRankStats	Immutable per-run observer statistics for Spiral ranking.
method	SpiralRankStats.limit_applied(self) -> bool	
method	SpiralRankStats.elapsed_ms(self) -> float	
method	SpiralRankStats.scoring_ms(self) -> float	
method	SpiralRankStats.sorting_ms(self) -> float	
method	SpiralRankStats.shaping_ms(self) -> float	
method	SpiralRankStats.to_dict(self) -> dict[str, Any]	
class	SpiralRankRecord	value model fields: trace_id, score, source_score, confidence, depth, age_ns, rank, native, config_id
method	SpiralRankRecord.to_dict(self) -> dict[str, Any]	
class	SpiralRankRun	Rank output plus immutable observer statistics.
method	SpiralRankRun.results(self) -> tuple[SpiralRankRecord, ...]	Backward-compatible alias for record rows, not a result envelope.
method	SpiralRankRun.to_dict(self) -> dict[str, Any]	
class	NativeSpiralRankEngine	Deterministic scorer/sorter for Spiral-compatible trace metadata.
method	NativeSpiralRankEngine.native_available(self) -> bool	
method	NativeSpiralRankEngine.score_many(self, scores: Iterable[float], confidences: Iterable[float] None=None, depths: Iterable[int] None=None, ages_ns: Iterable[int] None=None) -> list[float]	
method	NativeSpiralRankEngine.rank_run(self, trace_ids: Iterable[str], scores: Iterable[float], confidences: Iterable[float] None=None, depths: Iterable[int] None=None, ages_ns: Iterable[int] None=None, *, limit: int None=None, descending: bool=True) -> SpiralRankRun	

Type	Call / Class	Summary
method	<code>NativeSpiralRankEngine.rank(self, trace_ids: Iterable[str], scores: Iterable[float], confidences: Iterable[float] None=None, depths: Iterable[int] None=None, ages_ns: Iterable[int] None=None, *, limit: int None=None, descending: bool=True) -> list[SpiralRankRecord]</code>	
method	<code>NativeSpiralRankEngine.rank_result(self, trace_ids: Iterable[str], scores: Iterable[float], confidences: Iterable[float] None=None, depths: Iterable[int] None=None, ages_ns: Iterable[int] None=None, *, limit: int None=None, descending: bool=True) -> TDSResult</code>	AI-safe rank surface: always returns TDSResult, never raises.

staqtapp_tds.spiral.run

Type	Call / Class	Summary
function	<code>create_spiral_run(root, run_id: str, *, problem: Any None=None, problem_id: str='', description: str='', tags: tuple[str, ...]=())</code>	Create ``/spiral_runs/<run_id>/`` under a TDSDirectory.
class	<code>SpiralRunMetadata</code>	value model fields: run_id, problem_id, created_at, description, spiral_version, tags
method	<code>SpiralRunMetadata.to_dict(self) -> dict[str, Any]</code>	
class	<code>SpiralRun</code>	value model fields: run_dir, run_id
method	<code>SpiralRun.traces(self)</code>	
method	<code>SpiralRun.sets(self)</code>	
method	<code>SpiralRun.aggregations(self)</code>	
method	<code>SpiralRun.final(self)</code>	
method	<code>SpiralRun.metadata(self)</code>	
method	<code>SpiralRun.store_search_trace(self, trace_id: str, content: Any, *, rank_score: float None=None, rank_source: str='', created_by: str='', tags: tuple[str, ...]=()) -> TraceRecord</code>	
method	<code>SpiralRun.create_trace_set(self, set_id: str, trace_ids: list[str] tuple[str, ...], *, set_role: str='search_set', rank_policy: str='external', metadata: dict[str, Any] None=None) -> TraceSetManifest</code>	
method	<code>SpiralRun.store_aggregation(self, aggregation_id: str, output: Any, *, derived_from: list[str] tuple[str, ...], aggregation_step: int=1, rank_score: float None=None, rank_source: str='', metadata: dict[str, Any] None=None) -> AggregationRecord</code>	
method	<code>SpiralRun.store_final(self, name: str, output: Any, *, derived_from: list[str] tuple[str, ...]=()) -> None</code>	
method	<code>SpiralRun.ranked_traces(self, *, descending: bool=True) -> list[TraceRecord]</code>	
method	<code>SpiralRun.snapshot(self) -> dict[str, Any]</code>	

Support and utility surfaces

staqtapp_tds.arena

Type	Call / Class	Summary
class	<code>ArenaStats</code>	value model fields: capacity, used, free, allocations
class	<code>SharedMemoryArena</code>	Append-only byte arena returning int64 offset handles.
method	<code>SharedMemoryArena.capacity(self) -> int</code>	
method	<code>SharedMemoryArena.used(self) -> int</code>	
method	<code>SharedMemoryArena.free(self) -> int</code>	

Type	Call / Class	Summary
method	SharedMemoryArena.stats(self) -> ArenaStats	
method	SharedMemoryArena.allocate(self, payload: bytes bytearray memoryview) -> int	
method	SharedMemoryArena.read(self, handle: int) -> bytes	
method	SharedMemoryArena.view(self, handle: int) -> memoryview	Return a read-only-style memoryview slice. Caller must not mutate it.

staqtapp_tds.asi

Type	Call / Class	Summary
function	pressure_mode_for_score(score: int) -> PressureMode	
function	vfs_state_for_mode(mode: PressureMode) -> VFSState	
function	estimate_pressure(counters: Mapping[str, int], *, queue_depth: int=0, snapshot_lag: int=0, swiss_probe_pressure: int=0, radix_hot_prefix_score: int=0) -> PressureSnapshot	Compute a bounded ASI-storm pressure score from cheap counters only.
class	PressureMode	
method	PressureMode.label(self) -> str	
class	VFSState	
class	ChunkState	
class	PressureSnapshot	value model fields: score, mode, vfs_state, queue_depth, chunk_pending_count, chunk_quarantined_count, snapshot_lag, telemetry_dropped_rate, gil_reacquire_rate, allocator_pressure, radix_hot_prefix_score, swiss_probe_pressure
method	PressureSnapshot.to_dict(self) -> dict[str, object]	

staqtapp_tds.backends.python_index

Type	Call / Class	Summary
class	EntryIndexStats	value model fields: backend, size, shards, next_handle, capacity, tombstones, load_factor, max_probe, avg_probe
class	PythonEntryIndexBackend	
method	PythonEntryIndexBackend.put(self, key: str, entry: Any) -> int	
method	PythonEntryIndexBackend.get(self, key: str, default: Optional[Any]=None) -> Optional[Any]	
method	PythonEntryIndexBackend.get_handle(self, key: str) -> int	
method	PythonEntryIndexBackend.get_handles(self, keys: List[str]) -> List[int]	
method	PythonEntryIndexBackend.get_by_handle(self, handle: int) -> Optional[Any]	
method	PythonEntryIndexBackend.pop(self, key: str, default: Optional[Any]=None) -> Optional[Any]	
method	PythonEntryIndexBackend.keys(self) -> List[str]	
method	PythonEntryIndexBackend.values(self) -> List[Any]	
method	PythonEntryIndexBackend.items(self) -> List[Tuple[str, Any]]	
method	PythonEntryIndexBackend.contains(self, key: str) -> bool	
method	PythonEntryIndexBackend.stats(self) -> EntryIndexStats	

staqtapp_tds.capabilities

Type	Call / Class	Summary
class	ZoneCapability	
class	CapabilityRegistry	value model fields: flags
method	CapabilityRegistry.from_names(cls, names: Iterable[str]) -> 'CapabilityRegistry'	
method	CapabilityRegistry.enable(self, cap: ZoneCapability) -> None	
method	CapabilityRegistry.disable(self, cap: ZoneCapability) -> None	
method	CapabilityRegistry.supports(self, cap: ZoneCapability str) -> bool	
method	CapabilityRegistry.names(self) -> List[str]	
method	CapabilityRegistry.as_int(self) -> int	

staqtapp_tds.cluster

Type	Call / Class	Summary
function	query_requires_selector(**selectors) -> TDSResult	
class	TDSClusterIdentity	value model fields: name, schema_version, route_stamp_version, shards, created_ns, flags
method	TDSClusterIdentity.cluster_id(self) -> int	
method	TDSClusterIdentity.add_shard(self, path: str) -> None	
method	TDSClusterIdentity.feedback(self, *, entry_count: int=0, provenance_counts: Optional[Dict[str, int]]=None) -> Dict[str, object]	
method	TDSClusterIdentity.compact_record(self, *, entry_count: int=0, provenance_counts: Optional[Dict[str, int]]=None) -> np.ndarray	

staqtapp_tds.crypto

Type	Call / Class	Summary
class	CryptoProvider	
method	CryptoProvider.encrypt(self, data: bytes, key_id: str None=None) -> bytes	
method	CryptoProvider.decrypt(self, data: bytes, key_id: str None=None) -> bytes	
class	NoopCryptoProvider	
method	NoopCryptoProvider.encrypt(self, data: bytes, key_id: str None=None) -> bytes	
method	NoopCryptoProvider.decrypt(self, data: bytes, key_id: str None=None) -> bytes	
class	XorCryptoProvider	Tiny deterministic test provider, not production cryptography.
method	XorCryptoProvider.encrypt(self, data: bytes, key_id: str None=None) -> bytes	
method	XorCryptoProvider.decrypt(self, data: bytes, key_id: str None=None) -> bytes	

staqtapp_tds.errors

Type	Call / Class	Summary
class	ErrorLogMode	
class	ErrorRecord	value model fields: time_ns, code, path, name, severity
class	ErrorTelemetry	
method	ErrorTelemetry.record(self, code: str, *, path: str='', name: str='', severity: str='info') -> None	

Type	Call / Class	Summary
method	ErrorTelemetry.snapshot(self) -> Dict[str, object]	

staqtapp_tds.index

Type	Call / Class	Summary
class	EntryIndex	Native-ready entry index facade.
method	EntryIndex.native_status_result(self) -> TDSResult	Return the native load status for this EntryIndex without raising.
method	EntryIndex.backend_name(self) -> str	
method	EntryIndex.put(self, key: str, entry: Any) -> int	
method	EntryIndex.put_many(self, items: List[Tuple[str, Any]]) -> List[int]	
method	EntryIndex.get(self, key: str, default: Optional[Any]=None) -> Optional[Any]	
method	EntryIndex.get_handle(self, key: str) -> int	
method	EntryIndex.get_handles(self, keys: List[str]) -> List[int]	
method	EntryIndex.get_by_handle(self, handle: int) -> Optional[Any]	
method	EntryIndex.pop(self, key: str, default: Optional[Any]=None) -> Optional[Any]	
method	EntryIndex.pop_many(self, keys: List[str], default: Optional[Any]=None) -> List[Optional[Any]]	
method	EntryIndex.keys(self) -> List[str]	
method	EntryIndex.values(self) -> List[Any]	
method	EntryIndex.items(self) -> List[Tuple[str, Any]]	
method	EntryIndex.stats(self) -> Any	
method	EntryIndex.native_execution_stats(self) -> dict	

staqtapp_tds.invariants

Type	Call / Class	Summary
class	InvariantCode	
class	InvariantViolation	value model fields: code, path, name, observed, expected, severity, message
class	InvariantReport	value model fields: ok, path, checked, violations, numeric_records
method	InvariantReport.as_dict(self) -> Dict[str, Any]	
class	InvariantEngine	Consistency checks for a TDSDirectory.
method	InvariantEngine.evaluate_directory(self, directory: Any) -> InvariantReport	

staqtapp_tds.latency

Type	Call / Class	Summary
function	classify_latency(actual_ns: int, expected_ns: int, soft_limit_ns: int=0, hard_limit_ns: int=0) -> LatencyBucket	Classify a lookup duration using integer-only thresholds.
function	latency_ratio(actual_ns: int, expected_ns: int) -> float	
class	LatencyBucket	
class	LatencyPolicy	value model fields: expected_lookup_ns, soft_limit_ns, hard_limit_ns

Type	Call / Class	Summary
method	LatencyPolicy.classify(self, actual_ns: int) -> LatencyBucket	
method	LatencyPolicy.ratio(self, actual_ns: int) -> float	

staqtapp_tds.manifest

Type	Call / Class	Summary
function	find_manifest(start: Path) -> Optional[Path]	
function	load_manifest(folder: Path, *, inherit: bool=True) -> ManifestPolicy	
function	write_default_manifest(folder: Path, *, overwrite: bool=False) -> Path	
class	ManifestPolicy	value model fields: tds_manifest_version, folder_signature, schema_version, route_stamp_version, hash_policy, codec_policy, strict_mode, inherits, telemetry_mode, telemetry_flush_policy, trace_window, latency_policy
method	ManifestPolicy.to_dict(self, include_hash: bool=True) -> Dict[str, Any]	
method	ManifestPolicy.from_dict(cls, data: Dict[str, Any], *, inherited: Optional['ManifestPolicy']=None) -> 'ManifestPolicy'	
method	ManifestPolicy.default(cls) -> 'ManifestPolicy'	

staqtapp_tds.namespaces

Type	Call / Class	Summary
class	ReservedNamespaces	value model fields: directory_names, aliases, route_ids
method	ReservedNamespaces.from_dict(cls, data: Dict[str, object] None) -> 'ReservedNamespaces'	
method	ReservedNamespaces.to_dict(self) -> Dict[str, List[object]]	
method	ReservedNamespaces.is_reserved_directory(self, name: str) -> bool	
method	ReservedNamespaces.is_reserved_alias(self, alias: str) -> bool	
method	ReservedNamespaces.is_reserved_route_id(self, route_id: int) -> bool	
method	ReservedNamespaces.names(self) -> List[str]	
method	ReservedNamespaces.with_directory_names(self, names: Iterable[str]) -> 'ReservedNamespaces'	

staqtapp_tds.pressure

Type	Call / Class	Summary
function	calculate_pressure_snapshot(counters: Mapping[str, Any], *, native_counters: Mapping[str, Any] None=None, performance: Mapping[str, Any] None=None, storage: Mapping[str, Any] None=None, indexes: Mapping[str, Any] None=None, snapshot_lag: int=0) -> PressureCalculationSnapshot	Calculate multi-dimensional operational pressure from copied data only.
class	PressureComponent	One independently explainable pressure dimension.
method	PressureComponent.to_dict(self) -> dict[str, Any]	
class	PressureCalculationSnapshot	Dashboard-ready pressure interpretation assembled off the hot path.
method	PressureCalculationSnapshot.to_dict(self) -> dict[str, Any]	

staqtapp_tds.provenance

Type	Call / Class	Summary
function	stable_id(text: str) -> int	
class	ProvenanceClass	
class	ProvenanceTag	value model fields: provenance, source_id, source_uri, generation_id, transform_chain_hash, trust, flags, notes
method	ProvenanceTag.create(cls, provenance: str ProvenanceClass=ProvenanceClass.UNKNOWN, **kwargs) -> 'ProvenanceTag'	
method	ProvenanceTag.as_dict(self) -> Dict[str, object]	
method	ProvenanceTag.compact_record(self, entry_key: str) -> np.ndarray	

staqtapp_tds.radix

Type	Call / Class	Summary
class	RadixDirectoryRouter	Compressed-prefix router for direct children and slash paths.
method	RadixDirectoryRouter.get(self, key: str, default: Optional[T]=None) -> Optional[T]	
method	RadixDirectoryRouter.insert(self, key: str, value: T) -> None	
method	RadixDirectoryRouter.delete(self, key: str) -> Optional[T]	
method	RadixDirectoryRouter.keys(self) -> List[str]	
method	RadixDirectoryRouter.values(self) -> List[T]	
method	RadixDirectoryRouter.items(self) -> List[Tuple[str, T]]	
method	RadixDirectoryRouter.resolve_path(self, path: str) -> T	Resolve slash-separated child path through stored directory nodes.
method	RadixDirectoryRouter.lookup_steps(self, key: str) -> int	Return the number of compressed radix edges traversed for a lookup.
method	RadixDirectoryRouter.stats(self) -> dict	

staqtapp_tds.recovery

Type	Call / Class	Summary
function	build_recovery_plan(pressure: Mapping[str, Any] None, *, native_diagnostics: Mapping[str, Any] None=None, performance: Mapping[str, Any] None=None, storage: Mapping[str, Any] None=None) -> RecoveryPlan	Create an advisory recovery plan from immutable snapshot data only.
class	RecoveryAction	One safe, advisory recovery action.
method	RecoveryAction.to_dict(self) -> dict[str, Any]	
class	RecoveryPlan	Snapshot-ready Recovery Planner output.
method	RecoveryPlan.to_dict(self) -> dict[str, Any]	

staqtapp_tds.secure

Type	Call / Class	Summary
class	SecureParams	value model fields: values
method	SecureParams.from_env(cls, prefix: str='TDS_') -> 'SecureParams'	
method	SecureParams.from_mapping(cls, values: Mapping[str, str]) -> 'SecureParams'	
method	SecureParams.get_secret(self, name: str, default: str None=None) -> str None	
method	SecureParams.require(self, *names) -> None	
method	SecureParams.redact(self) -> dict[str, str]	

staqtapp_tds.serializers

Type	Call / Class	Summary
function	content_hash_bytes(raw: bytes, algorithm: str='sha256') -> str	
function	json_dumps_fast(value: Any) -> Tuple[bytes, str]	Return canonical JSON bytes and backend name via the centralized JSON layer.
function	json_loads_fast(raw: bytes) -> Tuple[Any, str]	Parse JSON bytes through the centralized simdjson-aware layer.
function	is_json_safe_value(value: Any) -> bool	
function	choose_variable_kind(value: Any) -> PayloadKind	
function	kind_name(kind: PayloadKind int) -> str	
class	PayloadKind	
class	CompressionPolicy	value model fields: enabled, codec, threshold_bytes
method	CompressionPolicy.should_compress(self, raw_len: int, *, force: bool None=None) -> bool	
class	SerializerDecision	value model fields: kind, fmt_name, json_backend

staqtapp_tds.srz

Type	Call / Class	Summary
function	route_id_for(stamp: str, path: str='') -> int	
class	SRZMetadata	value model fields: enabled, route_stamp, source_tags, aliases, latent_id, route_id, flags
method	SRZMetadata.create(cls, *, enabled: bool=False, route_stamp: str='', path: str='', source_tags: Iterable[str]=(), aliases: Iterable[str]=(), latent_id: Optional[int]=None, flags: int=0) -> 'SRZMetadata'	
method	SRZMetadata.as_dict(self) -> Dict[str, object]	
method	SRZMetadata.compact_record(self, *, dir_handle: int=-1, expected_ns: int=50000, avg_ns: int=0, hits: int=0, misses: int=0, bucket: int=0) -> np.ndarray	

staqtapp_tds.variables

Type	Call / Class	Summary
class	StalkState	value model fields: active, latest_index, latest_name, chain_names
class	VariableControl	
method	VariableControl.is_locked(self, name: str) -> bool	
method	VariableControl.addvar(self, name: str, data: Any) -> TDSResult	
method	VariableControl.editvar(self, name: str, data: Any, *, overwrite: bool=True) -> TDSResult	
method	VariableControl.lockvar(self, name: str, locked: bool=True) -> TDSResult	
method	VariableControl.unlockvar(self, name: str) -> TDSResult	
method	VariableControl.stalkvar(self, name: str, data: Any=None) -> TDSResult	
method	VariableControl.findvar(self, name: str) -> TDSResult	
method	VariableControl.loadvar(self, name: str) -> Any	
method	VariableControl.snapshot(self) -> Dict[str, Any]	
method	VariableControl.restore(self, snap: Dict[str, Any]) -> None	

staqtapp_tds.verify

Type	Call / Class	Summary
function	verify(target: Any) -> HealthReport	
class	HealthCheck	value model fields: name, status, detail, elapsed_ms
class	HealthReport	value model fields: status, score, generated_at, checks
method	HealthReport.to_dict(self) -> dict[str, Any]	
class	HealthVerifier	Explicit integrity verifier for TDS objects.
method	HealthVerifier.run(self) -> HealthReport	

Practical Operating Checklist

Use case	Preferred surface	Rule
AI driver proposal	Use <code>DriverFoundry.*</code>	Expect <code>DriverFoundryResult</code> ; never bypass Foundry for proposal/testing loops.
Production driver execution	Use <code>DriverRuntimeManager.execute_package</code>	Expect <code>DriverExecutionEvidence</code> ; Runtime Manager gates policy/registry/signature.
VM parity or harness testing	Use <code>DriverVMRuntime.execute</code> only under harness/manager control	Expect <code>DriverVMResult</code> ; closed/controlled, not open authority.
Storage operations	Use <code>*_result</code> surfaces where available	Expect <code>TDSResult</code> for recoverable conditions.
Browser telemetry	Use <code>AdminControl.status</code> and <code>/status.json</code>	Poll copied snapshots; no storage hot-path locking.
Driver Studio	Use <code>StudioQtBridge</code> and <code>live/hydrated</code> view models	Observe and prepare intent; no trust mutation.
Export verification	Use <code>Export/Audit</code> and <code>Export Integrity</code> workflows	Prepare deterministic evidence; do not sign/authorize.
Stress testing	Use regression/performance harnesses and result objects	Increase coverage, not authority.

Bottom line

The Driver VM runtime is a closed/controlled execution API. AI systems should interact through Foundry, Runtime Manager, regression/performance harnesses, storage `TDSResult` surfaces, and observer snapshots. Browser and Driver Studio can both run during `.tds` operations because they consume snapshots and bounded event streams rather than owning the `.tds` storage hot path.