

# Staqtapp-TDS v3.5.3

## Programmer Core API Guide - Release Supplement

This supplement is the authoritative v3.5.3 update to the broad v3.5.2 guide that follows. It makes the previously missing Phase 10 predecessor explicit, documents the Phase 11 safety corrections, and records the local release evidence without implying that remote cross-platform CI has already run.

### Phase progression

| Phase | Release meaning                | Completion evidence                                          |
|-------|--------------------------------|--------------------------------------------------------------|
| 9     | Incremental immutable segments | Content addressing, reuse, recovery, dry-run GC              |
| 10    | Controlled activation          | Explicit qualification, mode publication, verified rollback  |
| 11    | Release qualification          | GC adversarial tests, soak, docs, package and workflow gates |

### Safe default and authority boundary

If no storage mode record exists, **legacy TDSPersistence remains authoritative**. Qualification creates evidence but cannot activate the segmented path. Activation requires the exact acknowledgement `activate-guaranteed-segmented` and repeats the migration proof under the mutation lock immediately before atomic mode publication.

- No constructor, qualification call, Browser poll, or ordinary legacy commit can switch modes implicitly.
- A corrupt, non-canonical, incomplete, stale, or mismatched control record fails closed.
- Rollback reconstructs the current guaranteed bytes into a new verified legacy mount before authority changes.

### New public v3.5.3 types

`StorageMode`, `ControlledActivationError`, `ActivationQualification`, `StorageActivationStatus`, `ControlledCommitReport`, `ControlledStorage`, `SegmentReference`, `SegmentGenerationInfo`, `SegmentCommitReport`, `SegmentGCReport`, and `ImmutableSegmentStore` are exported from `staqtapp_tds`.

Detailed design and qualification records: `docs/118_v353_dev10_Controlled_Activation.md` and `docs/119_v353_dev11_Release_Qualification.md`.

### Current API index

Use `docs/reference/Programmers_API_Reference.md` for the current v3.5.3 storage API index. The separate `Staqtapp_TDS_API_Surface_Reference.pdf` is a historical v3.1.23 Driver/Studio reference and is not an exhaustive inventory of v3.5.3.

## Controlled activation operational contract

Construct the controller with separate guaranteed and legacy roots. First inspect the default mode, then qualify exact equivalence. Passing the qualification object alone is insufficient: the explicit acknowledgement is a separate operator-intent gate.

```
from staqtapp_tds import ControlledStorage, StorageMode

store = ControlledStorage(guaranteed_root, legacy_mount)
assert store.status().mode is StorageMode.LEGACY

qualification = store.qualify_activation()
assert qualification.activation_eligible

active = store.activate(
    qualification,
    acknowledgement=store.ACTIVATE_ACKNOWLEDGEMENT,
)
assert active.mode is StorageMode.GUARANTEED_SEGMENTED
assert active.current_generation_verified
```

### Mode-aware commit

`commit_filesystem(fs, parallel_nodes=True)` writes through the currently authorized path and returns `ControlledCommitReport`. The report identifies the mode, generation when applicable, archived file and source-byte counts, created/reused segments, and physical bytes written.

### Verified rollback

```
rolled = store.rollback_to_legacy(
    acknowledgement=store.ROLLBACK_ACKNOWLEDGEMENT,
)
assert rolled.mode is StorageMode.LEGACY
assert rolled.previous_mode is StorageMode.GUARANTEED_SEGMENTED
```

The rollback acknowledgement is exactly `rollback-to-legacy`. The original legacy mount is never overwritten. The current segmented generation is materialized to a private destination, its exact inventory and logical reopen behavior are verified, and only that new mount can become authoritative.

### Observation

`status(verify_current=True)` returns a bounded typed view containing mode, revision, qualification and current generation identifiers, activation verification, current-generation verification, rollback availability, and the authoritative legacy mount. Admin/Browser status exposes the same storage mode without becoming a control path.

**Operator rule. Treat qualification receipts and mode state as authority records. Do not edit them manually, reuse a qualification after source changes, or automate the acknowledgement strings as an implicit fallback.**

# Phase 11 GC and release qualification

## Closed-world segment collection

Destructive collection now inventories every recognized generation. If any generation is invalid or unreadable, collection is blocked and reports its identifier; public reference accounting also fails closed. For each candidate, reachability is recomputed twice and the regular-file type, device, inode, size, mode, mtime, and ctime are revalidated immediately before unlink.

```
from staqtapp_tds import ImmutableSegmentStore

segments = ImmutableSegmentStore(root)
preview = segments.collect_unreferenced_segments(dry_run=True)
if preview.blocked:
    raise RuntimeError(preview.invalid_generations)

# Destructive and explicit. Recomputes protection; does not trust preview.
report = segments.collect_unreferenced_segments(dry_run=False)
assert not report.blocked
```

- Changed candidates are retained and reported separately; only successful unlinks count toward removed bytes.
- The mutation lock covers scan, both rechecks, and delete; interrupted collection is idempotently resumable.
- Partially damaged generations preserve salvageable segments because incomplete evidence blocks deletion.

## Browser documentation evidence

The prior single Dashboard image and unsupported CSV description were removed. The repository now contains 19 distinct 1280 x 800 captures made by selecting every real Browser navigation control. Page 07 is the actual CSV Interpole Monitor after a real evidence chain reached Monitor Ready.

## Local release evidence

- Pure monolithic suite: 832 passed, 11 skipped (843 collected) in 55.07 seconds.
- Native-extension build/install and native-active monolithic suite: 843 passed in 55.89 seconds.
- Overlapping v3.5.3, workflow, Browser visual, and CSV monitor qualification group: 157 passed.
- PEP 517 wheel and sdist built; both passed twine metadata validation and archive-content audit.
- The isolated pure wheel exercised version, qualification, activation, rollback, and destructive GC successfully.

**Local qualification makes a review-branch push eligible. No push, tag, or publication is performed by this record. A v3.5.3 tag and PyPI publication remain prohibited until the configured Linux, Windows, macOS, Python 3.10-3.14, native, source-hygiene, and package jobs are green in GitHub Actions.**



# Staqtapp-TDS

## Programmer Core API Guide

Direct implementation calls for .tds storage, variables, trace ranking, CSV operations, Driver systems, Browser, and Driver Studio

|                  |                                                            |
|------------------|------------------------------------------------------------|
| Release          | v3.5.2                                                     |
| Audience         | AI-system programmers and integration engineers            |
| Primary contract | Result-first operations with explicit authority boundaries |
| Source basis     | Public v3.5.2 modules and validated usage tests            |

Use this guide first. The separate API Surface Reference remains the exhaustive class inventory.

---

# How to use this guide

This document is organized by programming task rather than by source-file order. Start with the quick recipes, then use the call maps for exact signatures. The API index includes public operational functions and the methods that perform storage, validation, execution, review, export, or observation. Passive dataclasses are introduced where programmers construct them, but their repetitive value-only properties are not expanded into noise.

## Three call styles

**Result-first:** use methods such as `write_result()`, `read_result()`, and `rank_trace_result()` in long-running AI services.

**Direct-value:** use `read_value()`, `read_text()`, and typed loaders when missing/corrupt data should be exceptional.

**Prepare / commit / replay:** CSV and semantic contracts make mutation authority explicit. Prepare is generally read-only; commit persists; replay detects drift.

All signatures in the reference sections are extracted from the v3.5.2 source tree used to build this PDF.

---

# Contents and PDF bookmarks

Use the document outline/bookmarks for immediate navigation. The major task-oriented sections are:

1. Ten-minute implementation path
2. Result-first programming model
3. Directory storage and entry operations
4. .tds persistence, mmap readers, and mounting
5. Variables, text, JSON, and serialization
6. Provenance, verification, telemetry, diagnostics, and recovery
7. Trace storage, provenance, and ranking
8. CSV Suite: direct operational workflow
9. CSV import, validation, export, and transactions
10. CSV scanning, row anchors, and artifact security
11. CSV storage bridge, native evidence, and performance gates
12. CSV Interpole and centralized Browser evidence
13. CSV Semantic IR candidates and lifecycle review
14. CSV operational call index
15. Driver TDDL, bytecode, Foundry, and Runtime Manager
16. Driver regression, review, evidence, and performance
17. Driver Studio and PyQt5 integration
18. Driver and Studio operational function index
19. Centralized Browser and local admin control
20. Configuration, native management, security, and low-level calls
21. Integration recipes
22. API selection matrix
23. Stable boundaries to preserve
24. Validation basis and compatibility
25. Final programmer checklist

# 1. Ten-minute implementation path

## 1.1 Store AI state and flush to .tds

```
from pathlib import Path
from staqtapp_tds import TDSFileSystem, TDSPersistence

fs = TDSFileSystem("agent_state")
runtime = fs.makedirs("/models/runtime")

runtime.write_text("system_prompt", "You are a careful planning agent.")
runtime.write_json("settings", {"temperature": 0.2, "tools": True})
runtime.write_result("step_count", 7)

result = runtime.read_result("settings")
if not result.ok:
    raise RuntimeError(f"{result.code}: {result.message}")

store = TDSPersistence(Path("./tds_store"))
store.flush(fs, parallel_nodes=False)

loaded = store.load_node(
    Path("./tds_store/agent_state__models__runtime.tds")
)
assert loaded.read_value("step_count") == 7
```

### Persistence naming

Each TDS directory node is persisted as its own .tds file. Nested directory names are joined with double underscores in the default persistence layout. Use the mapping returned by `TDSPersistence.flush()` or inspect the mount directory rather than manually guessing names in production code.

## 1.2 Controlled variables and stalk chains

```
state = fs.makedirs("/agent/state")

state.addvar("reward", 1.0)
state.editvar("reward", 1.25)
state.lockvar("reward")

found = state.findvar("reward")
assert found.ok and found.value == 1.25

state.unlockvar("reward")
state.addvar("context", ["initial"])
state.stalkvar("~context", ["observation-1"])
state.stalkvar("~context", ["observation-2"])

latest = state.loadvar("context_0002")
state.stalkvar("context") # clear tracked increments; keep the base
```

## 1.3 Rank trace candidates

```
from staqtapp_tds.spiral import rank_traces

ranked = rank_traces(
    ["trace-a", "trace-b", "trace-c"],
    [0.82, 0.95, 0.95],
    confidences=[0.90, 0.92, 0.92],
    depths=[2, 3, 1],
    limit=2,
)

for item in ranked:
    print(item.rank, item.trace_id, item.rank_score)
```

## 1.4 Import and validate CSV evidence

```
from staqtapp_tds.csv_layer import (
    export_original_csv,
    import_csv_bytes,
    prove_original_roundtrip,
    validate_csv_artifacts,
)

csv_dir = fs.makedirs("/datasets")
manifest = import_csv_bytes(
    csv_dir,
    b"id,name,score\n1,Ada,99\n2,Grace,98\n",
    source_name="people.csv",
)

report = validate_csv_artifacts(csv_dir, manifest.csv_id)
assert report.ok
assert prove_original_roundtrip(csv_dir, manifest.csv_id).byte_equivalent
original = export_original_csv(csv_dir, manifest.csv_id)
```

## 1.5 Start the centralized Browser

```
staqtapp-tds-admin status
staqtapp-tds-admin verify --sample
staqtapp-tds-admin serve-panel --host 127.0.0.1 --port 8765
```

Open <http://127.0.0.1:8765/>. The Browser reads snapshots and copied diagnostic events. It is not a polling loop over native index locks.

## 2. Result-first programming model

TDS distinguishes expected operational failures from programmer errors. Result-first calls return a `TDSResult` containing `ok`, `code`, `message`, `name`, `path`, `value`, and `meta`.

```
result = directory.read_result("agent_state")

if result.ok:
    value = result.value
else:
    logger.warning("TDS operation failed", extra={
        "code": str(result.code),
        "message": result.message,
        "path": result.path,
        "meta": result.meta,
    })
```

| Need                       | Preferred call                              | Behavior                                                                                  |
|----------------------------|---------------------------------------------|-------------------------------------------------------------------------------------------|
| Long-running service write | <code>TDSDirectory.write_result(...)</code> | Structured success/failure; no application-level exception required for normal conflicts. |
| Long-running service read  | <code>TDSDirectory.read_result(...)</code>  | Missing/corrupt data is represented in <code>TDSResult</code> .                           |
| Strict internal invariant  | <code>TDSDirectory.read_value(...)</code>   | Returns the value directly; missing entry raises.                                         |
| Human-readable text        | <code>TDSDirectory.read_text(...)</code>    | Returns str and validates the stored payload kind.                                        |
| Trace ranking evidence     | <code>rank_trace_result(...)</code>         | Returns ranked records inside <code>TDSResult</code> .                                    |
| Native engine inspection   | <code>native_status_result()</code>         | Availability/state is returned without halting the host service.                          |

### known\_result\_codes

```
known_result_codes() -> 'tuple[str, ...]'
```

**Import:** `staqtapp_tds.result`

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### is\_known\_result\_code

```
is_known_result_code(code: 'TDSResultCode | str') -> 'bool'
```

**Import:** `staqtapp_tds.result`

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### result\_info

```
result_info(code: 'TDSResultCode | str') -> 'TDSResultInfo | None'
```

**Import:** `staqtapp_tds.result`

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## 3. Directory storage and entry operations

The primary integration surface is `TDSFileSystem` plus the `TSDDirectory` returned by `fs.root`, `makedirs()`, `resolve()`, or `mkdir()`.

### TDSFileSystem

```
TDSFileSystem(name: 'str' = 'tds_root', manifest_policy: 'Optional[ManifestPolicy]' = None, runtime_config: 'Optional[Ru
```

Owns the directory tree, shared configuration, telemetry manager, and observation snapshot surface.

#### TDSFileSystem.resolve

```
resolve(self, path: 'str') -> 'TSDDirectory'
```

Resolve a directory path or raise when it does not exist.

#### TDSFileSystem.resolve\_radix

```
resolve_radix(self, path: 'str') -> 'TSDDirectory'
```

Public operational method on this integration surface.

#### TDSFileSystem.makedirs

```
makedirs(self, path: 'str', **kwargs) -> 'TSDDirectory'
```

Create an absolute or relative directory path and return the final `TSDDirectory`.

#### TDSFileSystem.parallel\_batch\_write

```
parallel_batch_write(self, writes: 'List[Tuple[str, str, Any]]') -> 'None'
```

Write multiple (path, name, value) items using the shared concurrency pool.

#### TDSFileSystem.snapshot\_headers

```
snapshot_headers(self) -> 'Dict[str, dict]'
```

Public operational method on this integration surface.

#### TDSFileSystem.telemetry\_snapshot

```
telemetry_snapshot(self) -> 'Dict[str, dict]'
```

Public operational method on this integration surface.

#### TDSFileSystem.observation\_snapshot

```
observation_snapshot(self, *, force: 'bool' = False) -> 'Dict[str, object]'
```

Public operational method on this integration surface.

#### TDSFileSystem.capability\_snapshot

```
capability_snapshot(self) -> 'Dict[str, List[str]]'
```

Public operational method on this integration surface.

### TSDDirectory

```
TSDDirectory(name: 'str', fmt_id: 'FmtID' = , flags: 'int' = , parent: "Optional['TSDDirectory']" = None, manifest_polic
```

Represents one directory node. Its write/read methods are the normal application API; low-level header/index helpers are advanced integration calls.

#### TSDDirectory.write

```
write(self, name: 'str', value: 'Any', fmt_id: 'FmtID' = , compress: 'bool | None' = False, codec: 'str' = '', provenanc
```

Store an entry in the directory.

#### TSDDirectory.write\_result

```
write_result(self, name: 'str', value: 'Any', fmt_id: 'FmtID' = , compress: 'bool | None' = False, codec: 'str' = '', pr
```

Store a value and return `TDSResult` instead of requiring exception-driven application control flow.

#### TSDDirectory.write\_variable

```
write_variable(self, name: 'str', value: 'Any', *, compress: 'bool | None' = None, codec: 'str' = '', provenance: 'Prove
```

Store the requested specialized payload and return the documented entry/result type.

#### TSDDirectory.write\_json

```
write_json(self, name: 'str', value: 'Any', *, overwrite: 'bool' = False, compress: 'bool | None' = None, codec: 'str' =
```

Store the requested specialized payload and return the documented entry/result type.

### **TDSDirectory.write\_text**

```
write_text(self, name: 'str', text: 'str', *, overwrite: 'bool' = False, compress: 'bool | None' = None, codec: 'str' =
```

Store the requested specialized payload and return the documented entry/result type.

### **TDSDirectory.write\_text\_chunked**

```
write_text_chunked(self, name: 'str', text: 'str', *, chunk_size: 'int' = 65536, overwrite: 'bool' = False, compress: 'b
```

Store large UTF-8 text as bounded chunks without cutting code points.

### **TDSDirectory.read**

```
read(self, name: 'str') -> 'TDSResult'
```

Read an entry.

### **TDSDirectory.read\_result**

```
read_result(self, name: 'str') -> 'TDSResult'
```

Return a TDSResult containing the value or a stable failure code.

### **TDSDirectory.read\_value**

```
read_value(self, name: 'str') -> 'Any'
```

Return the stored value directly; use when missing data should be exceptional.

### **TDSDirectory.read\_text**

```
read_text(self, name: 'str') -> 'str'
```

Read the requested specialized payload or metadata.

### **TDSDirectory.read\_text\_result**

```
read_text_result(self, name: 'str') -> 'TDSResult'
```

Read the requested specialized payload or metadata.

### **TDSDirectory.delete**

```
delete(self, name: 'str') -> 'TDSResult'
```

Delete an entry and return result evidence.

### **TDSDirectory.delete\_result**

```
delete_result(self, name: 'str') -> 'TDSResult'
```

Public operational method on this integration surface.

### **TDSDirectory.delete\_entry**

```
delete_entry(self, name: 'str') -> 'None'
```

Public operational method on this integration surface.

### **TDSDirectory.mkdir**

```
mkdir(self, name: 'str', **kwargs) -> "'TDSDirectory'"
```

Create a child directory.

### **TDSDirectory.cd**

```
cd(self, name: 'str') -> "'TDSDirectory'"
```

Public operational method on this integration surface.

### **TDSDirectory.ls**

```
ls(self, sort_by_prob: 'bool' = True) -> 'List[str]'
```

List entries/children.

### **TDSDirectory.parallel\_read\_all**

```
parallel_read_all(self) -> 'Dict[str, Any]'
```

Public operational method on this integration surface.

### **TDSDirectory.recursive\_join**

```
recursive_join(self, dtype=, max_depth: 'int' = 8) -> 'np.ndarray'
```

Public operational method on this integration surface.

### **TDSDirectory.build\_offset\_index**

```
build_offset_index(self) -> 'np.ndarray'
```

Public operational method on this integration surface.

### **TDSDirectory.entry\_metadata**

```
entry_metadata(self, name: 'str') -> 'dict'
```

Public operational method on this integration surface.

### **TDSDirectory.entry\_metadata\_result**

```
entry_metadata_result(self, name: 'str') -> 'TDSResult'
```

Public operational method on this integration surface.

### **TDSDirectory.invariant\_report**

```
invariant_report(self) -> 'dict'
```

Public operational method on this integration surface.

### **TDSDirectory.provenance\_record**

```
provenance_record(self, name: 'str') -> 'np.ndarray'
```

Public operational method on this integration surface.

### **TDSDirectory.provenance\_record\_result**

```
provenance_record_result(self, name: 'str') -> 'TDSResult'
```

Public operational method on this integration surface.

### **TDSDirectory.header\_bytes**

```
header_bytes(self) -> 'bytes'
```

Public operational method on this integration surface.

### **TDSDirectory.to\_bytes**

```
to_bytes(self) -> 'bytes'
```

Public operational method on this integration surface.

### **TDSDirectory.path**

```
path(self) -> 'str'
```

Public operational method on this integration surface.

### **TDSDirectory.telemetry\_snapshot**

```
telemetry_snapshot(self) -> 'dict'
```

Public operational method on this integration surface.

### **TDSDirectory.capability\_names**

```
capability_names(self) -> 'List[str]'
```

Public operational method on this integration surface.

### **TDSDirectory.is\_reserved\_namespace**

```
is_reserved_namespace(self, name: 'str') -> 'bool'
```

Public operational method on this integration surface.

### **TDSDirectory.reserved\_namespace\_names**

```
reserved_namespace_names(self) -> 'List[str]'
```

Public operational method on this integration surface.

### **TDSDirectory.srz\_record**

```
srz_record(self) -> 'np.ndarray'
```

Public operational method on this integration surface.

### **Format selection**

Use `write_text()` for text, `write_json()` for JSON-safe values, and `write_result()` or `write()` for general Python values. Restricted pickle compatibility remains behind the Serialization Manager and PicklePolicy boundary.

## 4. .tds persistence, mmap readers, and mounting

TDS persistence serializes each directory node into a compact .tds file with a fixed header, data block, slot index, and optional integrity sidecar. Writers use temporary files and atomic replacement. Readers validate file geometry, slot bounds, index hashes, metadata epoch, and content hashes before returning values.

### TDSPersistence

```
TDSPersistence(mount_dir, *, create_manifest: 'bool' = True)
```

High-level persistence manager for complete TDSFileSystem trees and mounted readers.

#### TDSPersistence.flush\_node

```
flush_node(self, node: 'TDSDirectory', parallel: 'bool' = False) -> 'Tuple[str, int]'
```

Public operational method on this integration surface.

#### TDSPersistence.flush

```
flush(self, fs: 'TDSFileSystem', parallel_nodes: 'bool' = True) -> 'Dict[str, int]'
```

Flush all filesystem nodes to .tds files using atomic replacement.

#### TDSPersistence.load\_node

```
load_node(self, tds_path, into: 'Optional[TDSDirectory]' = None) -> 'TDSDirectory'
```

Public operational method on this integration surface.

#### TDSPersistence.mount

```
mount(self, fs: 'TDSFileSystem') -> 'None'
```

Attach persisted readers to an existing filesystem for lazy direct reads.

#### TDSPersistence.unmount

```
unmount(self) -> 'Dict[str, int]'
```

Public operational method on this integration surface.

#### TDSPersistence.close\_reader

```
close_reader(self, tds_path) -> 'None'
```

Public operational method on this integration surface.

#### TDSPersistence.open\_readers

```
open_readers(self, paths) -> 'List[TDSReader]'
```

Public operational method on this integration surface.

### TDSWriter

```
TDSWriter(path)
```

Direct writer for one TDSDirectory and optional recursive descendants.

#### TDSWriter.write

```
write(self, directory: 'TDSDirectory', recurse: 'bool' = True) -> 'int'
```

Store an entry in the directory.

#### TDSWriter.write\_parallel

```
write_parallel(self, directory: 'TDSDirectory') -> 'int'
```

Store the requested specialized payload and return the documented entry/result type.

### TDSReader

```
TDSReader(path)
```

Validated mmap random-access reader for one .tds file.

#### TDSReader.reload

```
reload(self) -> 'None'
```

Public operational method on this integration surface.

## TDSReader.read

```
read(self, name: 'str', *, codec: 'str | None' = None, content_hash: 'str | None' = None) -> 'Any'
```

Read an entry.

## TDSReader.read\_raw

```
read_raw(self, name: 'str') -> 'bytes'
```

Return the stored raw slot payload bytes.

## TDSReader.read\_result

```
read_result(self, name: 'str') -> 'TDSResult'
```

Read the requested specialized payload or metadata.

## TDSReader.read\_many

```
read_many(self, names: 'List[str]') -> 'Dict[str, Any]'
```

Read the requested specialized payload or metadata.

## TDSReader.read\_many\_result

```
read_many_result(self, names: 'List[str]') -> 'TDSResult'
```

Read the requested specialized payload or metadata.

## TDSReader.keys

```
keys(self) -> 'List[str]'
```

Public operational method on this integration surface.

## TDSReader.close

```
close(self) -> 'None'
```

Release reader resources.

## ParallelFlusher

```
ParallelFlusher(mount_dir)
```

Queue multiple directory writes and complete them as a bounded batch.

## ParallelFlusher.enqueue

```
enqueue(self, node: 'TSDDirectory') -> 'None'
```

Public operational method on this integration surface.

## ParallelFlusher.flush\_all

```
flush_all(self, parallel: 'bool' = True) -> 'Dict[str, int]'
```

Complete all queued persistence writes.

```
from staqtapp_tds import TDSReader

reader = TDSReader("../tds_store/agent_state__models__runtime.tds")
try:
    # Tree flushes index entries by their fully qualified slot key.
    print(reader.keys())
    value = reader.read("/agent_state/models/runtime/settings")
    safe = reader.read_result(
        "/agent_state/models/runtime/missing_optional_value"
    )
finally:
    reader.close()
```

## Reader slot names

For files produced by a complete tree flush, `TDSReader.keys()` returns fully qualified slot keys such as `/agent_state/models/runtime/settings`. Use those keys for `read()` and `read_many()`.

`TDSPersistence.load_node()` reconstructs a directory and restores convenient short-name reads.

## Integrity failures

Header, index, slot-range, sidecar, epoch, codec, and content-hash failures are fail-closed. Do not bypass `TDSReader` by slicing the file directly unless you are implementing a separate forensic tool.

## 5. Variables, text, JSON, and serialization

### 5.1 VariableControl

#### VariableControl

```
VariableControl(directory: 'Any', *, error_mode: 'ErrorLogMode' = )
```

Can be created directly, but the same calls are exposed on TDSDirectory and are normally more convenient.

#### VariableControl.is\_locked

```
is_locked(self, name: 'str') -> 'bool'
```

Public operational method on this integration surface.

#### VariableControl.addvar

```
addvar(self, name: 'str', data: 'Any') -> 'TDSResult'
```

Public operational method on this integration surface.

#### VariableControl.editvar

```
editvar(self, name: 'str', data: 'Any', *, overwrite: 'bool' = True) -> 'TDSResult'
```

Public operational method on this integration surface.

#### VariableControl.lockvar

```
lockvar(self, name: 'str', locked: 'bool' = True) -> 'TDSResult'
```

Public operational method on this integration surface.

#### VariableControl.unlockvar

```
unlockvar(self, name: 'str') -> 'TDSResult'
```

Public operational method on this integration surface.

#### VariableControl.stalkvar

```
stalkvar(self, name: 'str', data: 'Any' = None) -> 'TDSResult'
```

Public operational method on this integration surface.

#### VariableControl.findvar

```
findvar(self, name: 'str') -> 'TDSResult'
```

Public operational method on this integration surface.

#### VariableControl.loadvar

```
loadvar(self, name: 'str') -> 'Any'
```

Public operational method on this integration surface.

#### VariableControl.snapshot

```
snapshot(self) -> 'Dict[str, Any]'
```

Return a JSON-safe snapshot.

#### VariableControl.restore

```
restore(self, snap: 'Dict[str, Any]') -> 'None'
```

Restore controller state from a prior snapshot.

### 5.2 Payload and serialization helpers

#### choose\_variable\_kind

```
choose_variable_kind(value: 'Any') -> 'PayloadKind'
```

**Import:** staqtapp\_tds.serializers

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### content\_hash\_bytes

```
content_hash_bytes(raw: 'bytes', algorithm: 'str' = 'sha256') -> 'str'
```

**Import:** staqtapp\_tds.serializers

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### **dumps\_pickle**

```
dumps_pickle(value: 'Any', policy: 'PicklePolicy | None' = None) -> 'bytes'
```

**Import:** staqtapp\_tds.tds\_pickle

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### **loads\_pickle**

```
loads_pickle(raw: 'bytes', policy: 'PicklePolicy | None' = None) -> 'Any'
```

**Import:** staqtapp\_tds.tds\_pickle

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### **pickle\_policy\_snapshot**

```
pickle_policy_snapshot(policy: 'PicklePolicy | None' = None) -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.tds\_pickle

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### **get\_default\_serialization\_manager**

```
get_default_serialization_manager() -> 'SerializationManager'
```

**Import:** staqtapp\_tds.serialization.manager

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## **SerializationManager**

```
SerializationManager(registry: 'CodecRegistry | None' = None)
```

First-class codec dispatch used by storage. Prefer directory write/read calls unless registering a custom codec or integrating an external transport.

### **SerializationManager.serialize**

```
serialize(self, value: 'Any', fmt_id: 'int | None' = None) -> 'EncodedPayload'
```

Public operational method on this integration surface.

### **SerializationManager.deserialize**

```
deserialize(self, raw: 'bytes', fmt_id: 'int') -> 'Any'
```

Public operational method on this integration surface.

### **SerializationManager.snapshot**

```
snapshot(self) -> 'dict[str, Any]'
```

Return a JSON-safe snapshot.

## **CodecRegistry**

```
CodecRegistry(codecs: 'Iterable[SerializationCodec]' = ())
```

Registry for SerializationCodec implementations.

### **CodecRegistry.register**

```
register(self, codec: 'SerializationCodec', *, replace: 'bool' = False) -> 'None'
```

Public operational method on this integration surface.

### **CodecRegistry.snapshot**

```
snapshot(self) -> 'dict[str, Any]'
```

Return a JSON-safe snapshot.

### **Pickle boundary**

General Python-object storage can use the restricted pickle codec for compatibility. Treat untrusted pickle bytes as hostile input. Prefer JSON, text, raw bytes, or a custom SerializationCodec for externally supplied data.

## 6. Provenance, verification, telemetry, diagnostics, and recovery

### 6.1 Health and invariant checks

#### verify

```
verify(target: 'Any') -> 'HealthReport'
```

**Import:** staqtapp\_tds.verify

**Behavior:** Run the TDS health verifier against a supported target and return a structured report.

#### estimate\_pressure

```
estimate_pressure(counters: 'Mapping[str, int]', *, queue_depth: 'int' = 0, snapshot_lag: 'int' = 0, swiss_probe_pressure: 'float' = 0.0) -> 'HealthReport'
```

**Import:** staqtapp\_tds.asi

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### calculate\_pressure\_snapshot

```
calculate_pressure_snapshot(counters: 'Mapping[str, Any]', *, native_counters: 'Mapping[str, Any] | None' = None, perform_snapshot: 'bool' = True) -> 'HealthReport'
```

**Import:** staqtapp\_tds.pressure

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### build\_recovery\_plan

```
build_recovery_plan(pressure: 'Mapping[str, Any] | None', *, native_diagnostics: 'Mapping[str, Any] | None' = None, perform_snapshot: 'bool' = True) -> 'HealthReport'
```

**Import:** staqtapp\_tds.recovery

**Behavior:** Convert pressure/diagnostic evidence into a non-mutating recovery plan.

#### classify\_latency

```
classify_latency(actual_ns: 'int', expected_ns: 'int', soft_limit_ns: 'int' = 0, hard_limit_ns: 'int' = 0) -> 'LatencyBudget'
```

**Import:** staqtapp\_tds.latency

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### latency\_ratio

```
latency_ratio(actual_ns: 'int', expected_ns: 'int') -> 'float'
```

**Import:** staqtapp\_tds.latency

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### InvariantEngine

```
InvariantEngine(*, max_entries: 'int' = -1, check_latency: 'bool' = False)
```

Run structural invariants and return reports without changing stored content.

### 6.2 Native diagnostics

#### native\_diagnostics\_available

```
native_diagnostics_available() -> 'bool'
```

**Import:** staqtapp\_tds.diagnostics

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### native\_diag\_snapshot

```
native_diag_snapshot(*, event_limit: 'int' = 32) -> 'NativeDiagnosticSnapshot'
```

**Import:** staqtapp\_tds.diagnostics

**Behavior:** Control or inspect the bounded native diagnostic side channel; not a storage data operation.

#### native\_diag\_reset

```
native_diag_reset() -> 'bool'
```

**Import:** staqtapp\_tds.diagnostics

**Behavior:** Control or inspect the bounded native diagnostic side channel; not a storage data operation.

### native\_diag\_set\_enabled

```
native_diag_set_enabled(enabled: 'bool') -> 'bool'
```

**Import:** staqtapp\_tds.diagnostics

**Behavior:** Control or inspect the bounded native diagnostic side channel; not a storage data operation.

### native\_diag\_mark\_degraded

```
native_diag_mark_degraded(degraded: 'bool' = True) -> 'bool'
```

**Import:** staqtapp\_tds.diagnostics

**Behavior:** Control or inspect the bounded native diagnostic side channel; not a storage data operation.

### native\_diag\_emit

```
native_diag_emit(event: 'DiagnosticEvent | int', value_a: 'int' = 0, value_b: 'int' = 0) -> 'bool'
```

**Import:** staqtapp\_tds.diagnostics

**Behavior:** Control or inspect the bounded native diagnostic side channel; not a storage data operation.

### native\_status\_result

```
native_status_result() -> 'TDSResult'
```

**Import:** staqtapp\_tds.native.manager

**Behavior:** Return a non-halting result describing native engine availability and state.

### native\_capabilities\_result

```
native_capabilities_result() -> 'TDSResult'
```

**Import:** staqtapp\_tds.native.manager

**Behavior:** Return a non-halting result containing the native capability matrix.

### get\_native\_manager

```
get_native_manager() -> 'NativeEngineManager'
```

**Import:** staqtapp\_tds.native.manager

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### Hot-path rule

Storage emits bounded counters/events; the diagnostic engine consumes copies. A full diagnostic ring drops diagnostic events and increments a counter rather than blocking storage.

## 6.3 Manifest, routing, and provenance helpers

### load\_manifest

```
load_manifest(folder: 'Path', *, inherit: 'bool' = True) -> 'ManifestPolicy'
```

**Import:** staqtapp\_tds.manifest

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### write\_default\_manifest

```
write_default_manifest(folder: 'Path', *, overwrite: 'bool' = False) -> 'Path'
```

**Import:** staqtapp\_tds.manifest

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### route\_id\_for

```
route_id_for(stamp: 'str', path: 'str' = '') -> 'int'
```

**Import:** staqtapp\_tds.srz

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### query\_requires\_selector

```
query_requires_selector(**selectors) -> 'TDSResult'
```

**Import:** staqtapp\_tds.cluster

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## 7. Trace storage, provenance, and ranking

The Spiral namespace is content-neutral: TDS stores and ranks trace evidence but does not implement an external reasoning system. Ranking is deterministic, supports native/Python parity, and can return records, a complete run, or TDSResult.

```
from staqtapp_tds.spiral import NativeSpiralRankEngine, SpiralRankConfig

engine = NativeSpiralRankEngine(
    SpiralRankConfig(),
    prefer_native=True,
)
ranked = engine.rank(
    ["trace-b", "trace-a", "trace-c"],
    [0.80, 0.91, 0.91],
    confidences=[0.90, 0.95, 0.95],
    depths=[1, 3, 1],
    ages_ns=[0, 0, 0],
)
assert [r.rank for r in ranked] == [1, 2, 3]
```

### create\_spiral\_run

```
create_spiral_run(root, run_id: 'str', *, problem: 'Any | None' = None, problem_id: 'str' = '', description: 'str' = '',
```

**Import:** staqtapp\_tds.spiral.run

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### rank\_traces

```
rank_traces(trace_ids: 'Iterable[str]', scores: 'Iterable[float]', **kwargs: 'Any') -> 'list[SpiralRankRecord]'
```

**Import:** staqtapp\_tds.spiral.rank

**Behavior:** Rank trace IDs deterministically from scores and optional confidence/depth/age inputs.

### rank\_trace\_run

```
rank_trace_run(trace_ids: 'Iterable[str]', scores: 'Iterable[float]', **kwargs: 'Any') -> 'SpiralRankRun'
```

**Import:** staqtapp\_tds.spiral.rank

**Behavior:** Return deterministic ranking output from the supplied trace evidence.

### rank\_trace\_result

```
rank_trace_result(trace_ids: 'Iterable[str]', scores: 'Iterable[float]', **kwargs: 'Any') -> 'TDSResult'
```

**Import:** staqtapp\_tds.spiral.rank

**Behavior:** Return deterministic ranking output from the supplied trace evidence.

## SpiralRun

```
SpiralRun(run_dir: 'Any', run_id: 'str') -> None
```

Stores one trace run under a TDS directory and maintains its manifest/provenance records.

### SpiralRun.metadata

```
metadata # property
```

Public operational method on this integration surface.

## NativeSpiralRankEngine

```
NativeSpiralRankEngine(config: 'SpiralRankConfig | None' = None, *, prefer_native: 'bool' = True)
```

Deterministic top-N ranking engine with optional compiled scoring.

### NativeSpiralRankEngine.native\_available

```
native_available # property
```

Public operational method on this integration surface.

### NativeSpiralRankEngine.score\_many

```
score_many(self, scores: 'Iterable[float]', confidences: 'Iterable[float] | None' = None, depths: 'Iterable[int] | None'
```

Public operational method on this integration surface.

---

### NativeSpiralRankEngine.rank

```
rank(self, trace_ids: 'Iterable[str]', scores: 'Iterable[float]', confidences: 'Iterable[float] | None' = None, depths:
```

Public operational method on this integration surface.

### NativeSpiralRankEngine.rank\_run

```
rank_run(self, trace_ids: 'Iterable[str]', scores: 'Iterable[float]', confidences: 'Iterable[float] | None' = None, dept
```

Public operational method on this integration surface.

### NativeSpiralRankEngine.rank\_result

```
rank_result(self, trace_ids: 'Iterable[str]', scores: 'Iterable[float]', confidences: 'Iterable[float] | None' = None, d
```

Public operational method on this integration surface.

## 8. CSV Suite: direct operational workflow

The CSV Suite is a layered evidence system. A normal integration does not need every layer. Choose the stopping point required by your application.

| Stage                   | Core calls                                                                                                                                       | When to stop                                                                       |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|
| Import/preserve         | <code>import_csv_bytes()</code> , <code>import_csv_file()</code>                                                                                 | You need managed CSV storage and original-byte identity.                           |
| Validate/reload/export  | <code>validate_csv_artifacts()</code> , <code>reload_csv_artifacts()</code> , <code>export_original_csv()</code>                                 | You need storage-grade artifact integrity and deterministic exports.               |
| Transaction/recovery    | <code>begin_csv_artifact_transaction()</code> , <code>commit_csv_artifact_transaction()</code> , <code>recover_csv_artifact_transaction()</code> | You need explicit staged publication and interrupted-write handling.               |
| Scan evidence           | <code>scan_csv_bytes()</code> , <code>scan_csv_row_anchors()</code> , <code>materialize_csv_scan_artifacts()</code>                              | You need deterministic row/anchor profiles.                                        |
| Storage/native evidence | <code>commit_csv_storage_bridge_manifest()</code> , <code>commit_csv_native_storage_artifacts()</code>                                           | You need complete storage-binding and native-storage evidence.                     |
| Kernel/Interpole        | <code>commit_csv_kernel_readiness_contract_report()</code> , <code>commit_csv_interpole_timeline_ring_report()</code>                            | You need native parity gates and comparative readiness telemetry.                  |
| Semantic review         | <code>prepare_csv_semantic_ir_candidate()</code> , <code>prepare_csv_semantic_ir_transition_batch()</code>                                       | You need explicit, authorized, read-only semantic propositions and review lineage. |

### CSV invariants

The preserved original remains the source evidence. Derived artifacts are reversible. The layer avoids per-cell storage writes. Optional native scan kernels must match the Python reference path. Semantic APIs never rewrite the CSV source.

## 9. CSV import, validation, export, and transactions

### 9.1 Core import/export calls

#### import\_csv\_bytes

```
import_csv_bytes(directory: 'TDSDirectory', raw: 'bytes', *, source_name: 'str' = 'csv_source.csv', encoding: 'str' = 'u
```

**Import:** staqtapp\_tds.csv\_layer.importer

**Behavior:** Import immutable CSV bytes and write the core CSV artifact family into a TDS directory.

#### import\_csv\_file

```
import_csv_file(directory: 'TDSDirectory', path: 'str | Path', *, encoding: 'str' = 'utf-8', csv_id: 'str | None' = None
```

**Import:** staqtapp\_tds.csv\_layer.importer

**Behavior:** Read a file, preserve its bytes/text identity, and write the core CSV artifact family.

#### load\_csv\_manifest

```
load_csv_manifest(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVImportManifest'
```

**Import:** staqtapp\_tds.csv\_layer.importer

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

#### validate\_csv\_artifacts

```
validate_csv_artifacts(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVArtifactValidationReport'
```

**Import:** staqtapp\_tds.csv\_layer.validator

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

#### reload\_csv\_artifacts

```
reload_csv_artifacts(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVReloadedArtifacts'
```

**Import:** staqtapp\_tds.csv\_layer.reload

**Behavior:** Reload the committed CSV artifact family into typed value objects.

#### export\_original\_csv

```
export_original_csv(directory: 'TDSDirectory', csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.exporter

**Behavior:** Return the preserved original CSV text without canonical rewriting.

#### export\_canonical\_csv

```
export_canonical_csv(directory: 'TDSDirectory', csv_id: 'str', *, lineterminator: 'str' = '\n') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.exporter

**Behavior:** Return a deterministic canonical CSV rendering while leaving the preserved source unchanged.

#### prove\_original\_roundtrip

```
prove_original_roundtrip(directory: 'TDSDirectory', csv_id: 'str', *, overwrite: 'bool' = True) -> 'CSVRoundTripReport'
```

**Import:** staqtapp\_tds.csv\_layer.exporter

**Behavior:** Write and return original-byte round-trip proof.

#### load\_csv\_artifact

```
load_csv_artifact(directory: 'TDSDirectory', csv_id: 'str', artifact: 'str') -> 'Any'
```

**Import:** staqtapp\_tds.csv\_layer.exporter

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

#### artifact\_keys

```
artifact_keys(csv_id: 'str') -> 'dict[str, str]'
```

**Import:** staqtapp\_tds.csv\_layer.manifest

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### 9.2 Transaction and recovery calls

#### new\_csv\_transaction\_id

```
new_csv_transaction_id() -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### begin\_csv\_artifact\_transaction

```
begin_csv_artifact_transaction(directory: 'TDSDirectory', raw: 'bytes', *, source_name: 'str' = 'csv_source.csv', encoding: 'str' = 'utf-8')
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Stage a complete core CSV artifact family under a transaction namespace.

### validate\_csv\_artifact\_transaction

```
validate_csv_artifact_transaction(directory: 'TDSDirectory', csv_id: 'str', transaction_id: 'str') -> 'CSVArtifactTransactionReport'
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### commit\_csv\_artifact\_transaction

```
commit_csv_artifact_transaction(directory: 'TDSDirectory', csv_id: 'str', transaction_id: 'str', *, overwrite: 'bool' = False)
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Validate a staged CSV transaction and publish the complete artifact family.

### recover\_csv\_artifact\_transaction

```
recover_csv_artifact_transaction(directory: 'TDSDirectory', csv_id: 'str', transaction_id: 'str', *, commit_staged: 'bool' = False)
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Inspect or recover an interrupted CSV artifact transaction.

### detect\_partial\_csv\_artifacts

```
detect_partial_csv_artifacts(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVArtifactTransactionReport'
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Detect the requested condition or format and return deterministic evidence.

### load\_csv\_artifact\_transaction\_report

```
load_csv_artifact_transaction_report(directory: 'TDSDirectory', csv_id: 'str', transaction_id: 'str | None' = None) -> 'CSVArtifactTransactionReport'
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### csv\_artifact\_transaction\_keys

```
csv_artifact_transaction_keys(csv_id: 'str', transaction_id: 'str') -> 'dict[str, str]'
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### validate\_csv\_transaction\_id

```
validate_csv_transaction_id(transaction_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

```
from staqtapp_tds.csv_layer import (
    begin_csv_artifact_transaction,
    commit_csv_artifact_transaction,
    validate_csv_artifact_transaction,
)

staged = begin_csv_artifact_transaction(
    csv_dir,
    b"id,note\n1,alpha\n2,beta\n",
    source_name="events.csv",
    transaction_id="tx-events-001",
)
assert staged.ok and staged.status == "staged"

checked = validate_csv_artifact_transaction(
    csv_dir, staged.csv_id, staged.transaction_id
)
assert checked.ok

committed = commit_csv_artifact_transaction(
    csv_dir, staged.csv_id, staged.transaction_id,
    overwrite=True,
```

---

```
    csv_dir, staged.csv_id, staged.transaction_id
  )
  assert committed.ok and committed.status == "committed"
```

## 10. CSV scanning, row anchors, and artifact security

### detect\_csv\_dialect

```
detect_csv_dialect(text: 'str', *, delimiters: 'Iterable[str]' = (',', ';', '\t', '|')) -> 'CSVDialectFingerprint'
```

**Import:** staqtapp\_tds.csv\_layer.dialect

**Behavior:** Detect the requested condition or format and return deterministic evidence.

### dialect\_to\_csv\_kwargs

```
dialect_to_csv_kwargs(dialect: 'CSVDialectFingerprint') -> 'dict[str, object]'
```

**Import:** staqtapp\_tds.csv\_layer.dialect

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### logical\_record\_offsets

```
logical_record_offsets(text: 'str', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8') -> 'tuple[int, ...]'
```

**Import:** staqtapp\_tds.csv\_layer.row\_offsets

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### logical\_record\_offsets\_bytes

```
logical_record_offsets_bytes(raw: 'bytes | bytearray | memoryview', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8') -> 'tuple[int, ...]'
```

**Import:** staqtapp\_tds.csv\_layer.row\_offsets

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### build\_row\_offset\_map

```
build_row_offset_map(text: 'str', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8', raw: 'bytes | None' = None) -> 'dict[int, int]'
```

**Import:** staqtapp\_tds.csv\_layer.row\_offsets

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### pack\_csv\_row\_offsets

```
pack_csv_row_offsets(row_offsets: 'Iterable[int]') -> 'bytes'
```

**Import:** staqtapp\_tds.csv\_layer.row\_offsets

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### unpack\_csv\_row\_offsets

```
unpack_csv_row_offsets(payload: 'bytes | bytearray | memoryview') -> 'tuple[int, ...]'
```

**Import:** staqtapp\_tds.csv\_layer.row\_offsets

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### scan\_csv\_bytes

```
scan_csv_bytes(raw: 'BytesLike', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8', chunk_size: 'int | None' = None) -> 'dict[str, dict[str, int]]'
```

**Import:** staqtapp\_tds.csv\_layer.scanner

**Behavior:** Build a deterministic reference scan profile from immutable CSV bytes.

### scan\_csv\_text

```
scan_csv_text(text: 'str', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8', chunk_size: 'int | None' = None) -> 'dict[str, dict[str, int]]'
```

**Import:** staqtapp\_tds.csv\_layer.scanner

**Behavior:** Scan immutable input and return a deterministic profile or ranking result.

### validate\_csv\_scan\_profile

```
validate_csv_scan_profile(directory: 'TSDDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVScanParity'
```

**Import:** staqtapp\_tds.csv\_layer.scanner

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### scan\_csv\_row\_anchors

```
scan_csv_row_anchors(raw: 'BytesLike', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8', chunk_size: 'int | None' = None) -> 'dict[str, dict[str, int]]'
```

**Import:** staqtapp\_tds.csv\_layer.scanner

**Behavior:** Build row-anchor hashes and profile information from immutable CSV bytes.

### scan\_csv\_text\_row\_anchors

```
scan_csv_text_row_anchors(text: 'str', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8', chunk_size: 'int
```

**Import:** staqtapp\_tds.csv\_layer.scanner

**Behavior:** Scan immutable input and return a deterministic profile or ranking result.

### validate\_csv\_row\_anchors

```
validate_csv_row_anchors(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVRowAnchorP
```

**Import:** staqtapp\_tds.csv\_layer.scanner

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### csv\_scan\_artifact\_keys

```
csv_scan_artifact_keys(csv_id: 'str') -> 'dict[str, str]'
```

**Import:** staqtapp\_tds.csv\_layer.scan\_artifacts

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### materialize\_csv\_scan\_artifacts

```
materialize_csv_scan_artifacts(directory: 'TSDirectory', csv_id: 'str', *, include_row_anchors: 'bool' = True, chunk_si
```

**Import:** staqtapp\_tds.csv\_layer.scan\_artifacts

**Behavior:** Persist scan profile and row-anchor evidence as bounded derived artifacts.

### validate\_materialized\_csv\_scan\_artifacts

```
validate_materialized_csv_scan_artifacts(directory: 'TSDirectory', csv_id: 'str', *, require_row_anchors: 'bool' = True
```

**Import:** staqtapp\_tds.csv\_layer.scan\_artifacts

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### load\_csv\_scan\_profile

```
load_csv_scan_profile(directory: 'TSDirectory', csv_id: 'str') -> 'CSVScanProfile'
```

**Import:** staqtapp\_tds.csv\_layer.scan\_artifacts

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### load\_csv\_row\_anchor\_profile

```
load_csv_row_anchor_profile(directory: 'TSDirectory', csv_id: 'str') -> 'CSVRowAnchorProfile'
```

**Import:** staqtapp\_tds.csv\_layer.scan\_artifacts

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### load\_csv\_scan\_materialization\_report

```
load_csv_scan_materialization_report(directory: 'TSDirectory', csv_id: 'str') -> 'CSVScanArtifactReport'
```

**Import:** staqtapp\_tds.csv\_layer.scan\_artifacts

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_artifact\_key

```
validate_csv_artifact_key(key: 'str', csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.security

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### validate\_csv\_artifact\_security

```
validate_csv_artifact_security(directory: 'TSDirectory', csv_id: 'str', *, include_scan_artifacts: 'bool' = False) -> '
```

**Import:** staqtapp\_tds.csv\_layer.security

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

# 11. CSV storage bridge, native evidence, and performance gates

## csv\_storage\_bridge\_plan

```
csv_storage_bridge_plan(csv_id: 'str', *, include_scan_artifacts: 'bool' = False, include_transaction_report: 'bool' = False)
```

**Import:** staqtapp\_tds.csv\_layer.storage\_bridge

**Behavior:** Return the deterministic operation plan without performing storage writes.

## validate\_csv\_storage\_bridge\_preflight

```
validate_csv_storage_bridge_preflight(directory: 'TDSDirectory', csv_id: 'str', *, include_scan_artifacts: 'bool' = False)
```

**Import:** staqtapp\_tds.csv\_layer.storage\_bridge

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## prepare\_csv\_storage\_bridge\_commit

```
prepare_csv_storage_bridge_commit(directory: 'TDSDirectory', csv_id: 'str', *, include_scan_artifacts: 'bool' = False, r...
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

## commit\_csv\_storage\_bridge\_manifest

```
commit_csv_storage_bridge_manifest(directory: 'TDSDirectory', csv_id: 'str', *, include_scan_artifacts: 'bool' = False, r...
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

## load\_csv\_storage\_bridge\_commit\_report

```
load_csv_storage_bridge_commit_report(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVStorageBridgeCommitReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

## validate\_csv\_storage\_bridge\_commit

```
validate_csv_storage_bridge_commit(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVStorageBridgeCommitReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## prepare\_csv\_storage\_adapter\_binding

```
prepare_csv_storage_adapter_binding(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVStorageAdapterBinding'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

## validate\_csv\_storage\_adapter\_binding

```
validate_csv_storage_adapter_binding(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVStorageAdapterBinding'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## prepare\_csv\_storage\_adapter\_replay

```
prepare_csv_storage_adapter_replay(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVStorageAdapterReplayReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

## commit\_csv\_storage\_adapter\_replay\_report

```
commit_csv_storage_adapter_replay_report(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, r...
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

## load\_csv\_storage\_adapter\_replay\_report

```
load_csv_storage_adapter_replay_report(directory: 'TSDirectory', csv_id: 'str') -> 'CSVStorageAdapterReplayReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

## validate\_csv\_storage\_adapter\_replay

```
validate_csv_storage_adapter_replay(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVStorageAdapterReplayReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## commit\_csv\_native\_storage\_artifacts

```
commit_csv_native_storage_artifacts(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, overwrite: bool = False)
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

## load\_csv\_native\_storage\_commit\_report

```
load_csv_native_storage_commit_report(directory: 'TSDirectory', csv_id: 'str') -> 'CSVNativeStorageCommitReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

## validate\_csv\_native\_storage\_commit

```
validate_csv_native_storage_commit(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: str = 'utf-8')
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## prepare\_csv\_native\_storage\_revalidation

```
prepare_csv_native_storage_revalidation(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: str = 'utf-8')
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

## commit\_csv\_native\_storage\_revalidation\_report

```
commit_csv_native_storage_revalidation_report(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: str = 'utf-8')
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

## load\_csv\_native\_storage\_revalidation\_report

```
load_csv_native_storage_revalidation_report(directory: 'TSDirectory', csv_id: 'str') -> 'CSVNativeStorageRevalidationReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

## validate\_csv\_native\_storage\_revalidation

```
validate_csv_native_storage_revalidation(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: str = 'utf-8')
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## prepare\_csv\_kernel\_readiness\_contract

```
prepare_csv_kernel_readiness_contract(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7) -> 'CSVKernelReadinessContract'
```

**Import:** staqtapp\_tds.csv\_layer.kernel

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

## commit\_csv\_kernel\_readiness\_contract\_report

```
commit_csv_kernel_readiness_contract_report(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7, encoding: str = 'utf-8')
```

**Import:** staqtapp\_tds.csv\_layer.kernel

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

## load\_csv\_kernel\_readiness\_contract\_report

```
load_csv_kernel_readiness_contract_report(directory: 'TSDirectory', csv_id: 'str') -> 'CSVKernelReadinessReport'
```

**Import:** staqtapp\_tds.csv\_layer.kernel

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_kernel\_readiness\_contract

```
validate_csv_kernel_readiness_contract(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7) -> 'CS'
```

**Import:** staqtapp\_tds.csv\_layer.kernel

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### prepare\_csv\_native\_scan\_kernel\_prototype

```
prepare_csv_native_scan_kernel_prototype(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7, use_
```

**Import:** staqtapp\_tds.csv\_layer.native\_scan

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### commit\_csv\_native\_scan\_kernel\_prototype\_report

```
commit_csv_native_scan_kernel_prototype_report(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7
```

**Import:** staqtapp\_tds.csv\_layer.native\_scan

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### load\_csv\_native\_scan\_kernel\_prototype\_report

```
load_csv_native_scan_kernel_prototype_report(directory: 'TSDirectory', csv_id: 'str') -> 'CSVNativeScanKernelReport'
```

**Import:** staqtapp\_tds.csv\_layer.native\_scan

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_native\_scan\_kernel\_prototype

```
validate_csv_native_scan_kernel_prototype(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7) ->
```

**Import:** staqtapp\_tds.csv\_layer.native\_scan

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### prepare\_csv\_native\_row\_anchor\_kernel

```
prepare_csv_native_row_anchor_kernel(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7, use_nati
```

**Import:** staqtapp\_tds.csv\_layer.native\_row\_anchor

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### commit\_csv\_native\_row\_anchor\_kernel\_report

```
commit_csv_native_row_anchor_kernel_report(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7, us
```

**Import:** staqtapp\_tds.csv\_layer.native\_row\_anchor

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### load\_csv\_native\_row\_anchor\_kernel\_report

```
load_csv_native_row_anchor_kernel_report(directory: 'TSDirectory', csv_id: 'str') -> 'CSVNativeRowAnchorKernelReport'
```

**Import:** staqtapp\_tds.csv\_layer.native\_row\_anchor

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_native\_row\_anchor\_kernel

```
validate_csv_native_row_anchor_kernel(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7) -> 'CSV
```

**Import:** staqtapp\_tds.csv\_layer.native\_row\_anchor

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### prepare\_csv\_kernel\_performance\_gates

```
prepare_csv_kernel_performance_gates(directory: 'TSDirectory', csv_id: 'str', *, max_report_json_bytes: 'int' = 65536,
```

**Import:** staqtapp\_tds.csv\_layer.performance\_gates

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### commit\_csv\_kernel\_performance\_gate\_report

```
commit_csv_kernel_performance_gate_report(directory: 'TSDirectory', csv_id: 'str', *, max_report_json_bytes: 'int' = 65
```

**Import:** staqtapp\_tds.csv\_layer.performance\_gates

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

---

### load\_csv\_kernel\_performance\_gate\_report

```
load_csv_kernel_performance_gate_report(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVKernelPerformanceGateReport'
```

**Import:** staqtapp\_tds.csv\_layer.performance\_gates

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_kernel\_performance\_gate\_report

```
validate_csv_kernel_performance_gate_report(directory: 'TDSDirectory', csv_id: 'str', *, max_report_json_bytes: 'int | None')
```

**Import:** staqtapp\_tds.csv\_layer.performance\_gates

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract

#### Native mode

Use use\_native=True to request the optional sidecar and force\_native=True only when absence or mismatch must fail the operation. Default-safe operation retains the reference Python path.

## 12. CSV Interpole and centralized Browser evidence

### prepare\_csv\_interpole\_timeline

```
prepare_csv_interpole_timeline(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: 'e
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### commit\_csv\_interpole\_timeline\_report

```
commit_csv_interpole_timeline_report(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, overw
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### load\_csv\_interpole\_timeline\_report

```
load_csv_interpole_timeline_report(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVInterpoleTimelineReport'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_interpole\_timeline

```
validate_csv_interpole_timeline(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: 'e
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### csv\_interpole\_timeline\_summary

```
csv_interpole_timeline_summary(report: 'CSVInterpoleTimelineReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### prepare\_csv\_interpole\_determinant\_vector

```
prepare_csv_interpole_determinant_vector(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, e
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### commit\_csv\_interpole\_determinant\_vector\_report

```
commit_csv_interpole_determinant_vector_report(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = N
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### load\_csv\_interpole\_determinant\_vector\_report

```
load_csv_interpole_determinant_vector_report(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVInterpoleDeterminantVector'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_interpole\_determinant\_vector

```
validate_csv_interpole_determinant_vector(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None,
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### csv\_interpole\_determinant\_vector\_summary

```
csv_interpole_determinant_vector_summary(report: 'CSVInterpoleDeterminantVectorReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### prepare\_csv\_interpole\_timeline\_ring

```
prepare_csv_interpole_timeline_ring(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encodi
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

## commit\_csv\_interpole\_timeline\_ring\_report

```
commit_csv_interpole_timeline_ring_report(directory: 'TSDDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None,
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

## load\_csv\_interpole\_timeline\_ring\_report

```
load_csv_interpole_timeline_ring_report(directory: 'TSDDirectory', csv_id: 'str') -> 'CSVInterpoleTimelineRingReport'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

## validate\_csv\_interpole\_timeline\_ring

```
validate_csv_interpole_timeline_ring(directory: 'TSDDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encod
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## csv\_interpole\_timeline\_ring\_summary

```
csv_interpole_timeline_ring_summary(report: 'CSVInterpoleTimelineRingReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

## prepare\_csv\_interpole\_browser\_monitor\_snapshot

```
prepare_csv_interpole_browser_monitor_snapshot(directory: 'TSDDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7
```

**Import:** staqtapp\_tds.csv\_layer.browser\_monitor

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

## validate\_csv\_interpole\_browser\_monitor\_snapshot

```
validate_csv_interpole_browser_monitor_snapshot(snapshot: 'CSVInterpoleBrowserMonitorSnapshot | Mapping[str, Any]', *, p
```

**Import:** staqtapp\_tds.csv\_layer.browser\_monitor

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## replay\_csv\_interpole\_browser\_monitor\_snapshot

```
replay_csv_interpole_browser_monitor_snapshot(directory: 'TSDDirectory', csv_id: 'str', source_snapshot: 'CSVInterpoleBr
```

**Import:** staqtapp\_tds.csv\_layer.browser\_monitor

**Behavior:** Recompute the current result and compare it with the supplied source object to detect drift or tampering.

## csv\_interpole\_browser\_monitor\_summary

```
csv_interpole_browser_monitor_summary(snapshot: 'CSVInterpoleBrowserMonitorSnapshot') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.browser\_monitor

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

## csv\_interpole\_browser\_monitor\_replay\_summary

```
csv_interpole_browser_monitor_replay_summary(report: 'CSVInterpoleMonitorReplayReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.browser\_monitor

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

Interpole reports comparative movement, determinant vectors, timeline-ring evidence, mirror feedback, and readiness pressure. They do not infer schemas, entities, or semantic truth. The Browser monitor is read-only and bounded.

## 13. CSV Semantic IR candidates and lifecycle review

Semantic IR starts only after an explicit caller opt-in. Declarations are caller-authored propositions bound to named evidence. TDS validates contract shape, source identity, handoff evidence, fingerprints, replay, authorization scope, and lifecycle ordering.

```
from staqtapp_tds.csv_layer import (
    CSVSemanticIRDeclaration,
    CSVSemanticIRTransitionAuthorization,
    CSVSemanticIRTransitionRequest,
    prepare_csv_semantic_ir_candidate,
    prepare_csv_semantic_ir_transition,
)

declaration = CSVSemanticIRDeclaration(
    proposition_id="customer_identifier_role",
    semantic_kind="column_role",
    subject_scope="column",
    subject_locator="index:0",
    predicate="represents",
    object_value="CustomerIdentifier",
    evidence_names=("core_artifact_integrity", "native_row_anchor_parity"),
)

candidate = prepare_csv_semantic_ir_candidate(
    csv_dir,
    manifest.csv_id,
    (declaration,),
    explicit_opt_in=True,
)

request = CSVSemanticIRTransitionRequest(
    transition_id="transition-001",
    proposition_id="customer_identifier_role",
    from_state="proposed",
    to_state="validated",
    reason="Reviewer confirmed the proposition against referenced evidence.",
    authorization=CSVSemanticIRTransitionAuthorization(
        authorization_id="auth-001",
        actor_id="reviewer-001",
        authority_scope="validate_proposition",
        authorization_reference="ticket:SEM-001",
    ),
)

lifecycle = prepare_csv_semantic_ir_transition(
    csv_dir, manifest.csv_id, candidate, request
)
```

### Prerequisite

A candidate requires the complete validated Semantic IR handoff evidence chain. The short snippet focuses on object construction; build and validate the storage/native/Interpole/kernel reports first, or use an application helper that prepares that chain.

### prepare\_csv\_semantic\_ir\_handoff

```
prepare_csv_semantic_ir_handoff(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7, source_monitor: 'SourceMonitor' = None) -> 'CSVSemanticIRHandoffReport'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_handoff

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### validate\_csv\_semantic\_ir\_handoff

```
validate_csv_semantic_ir_handoff(report: 'CSVSemanticIRHandoffReport | Mapping[str, Any]') -> 'CSVSemanticIRHandoffValidatedReport'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_handoff

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### csv\_semantic\_ir\_handoff\_fingerprint

```
csv_semantic_ir_handoff_fingerprint(report: 'CSVSemanticIRHandoffReport | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_handoff

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### csv\_semantic\_ir\_handoff\_summary

```
csv_semantic_ir_handoff_summary(report: 'CSVSemanticIRHandoffReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_handoff

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### prepare\_csv\_semantic\_ir\_candidate

```
prepare_csv_semantic_ir_candidate(directory: 'TDSDirectory', csv_id: 'str', declarations: 'Sequence[CSVSemanticIRDeclaration]') -> 'CSVSemanticIRCandidate | Mapping[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Create an in-memory, caller-declared Semantic IR candidate after explicit opt-in and evidence validation.

### validate\_csv\_semantic\_ir\_candidate

```
validate_csv_semantic_ir_candidate(candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'CSVSemanticIRValidationReport | Mapping[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### replay\_csv\_semantic\_ir\_candidate

```
replay_csv_semantic_ir_candidate(directory: 'TDSDirectory', csv_id: 'str', source_candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'CSVSemanticIRCandidate | Mapping[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Recompute the current result and compare it with the supplied source object to detect drift or tampering.

### csv\_semantic\_ir\_candidate\_fingerprint

```
csv_semantic_ir_candidate_fingerprint(candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### csv\_semantic\_ir\_candidate\_summary

```
csv_semantic_ir_candidate_summary(candidate: 'CSVSemanticIRCandidate') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### csv\_semantic\_ir\_replay\_summary

```
csv_semantic_ir_replay_summary(report: 'CSVSemanticIRReplayReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### prepare\_csv\_semantic\_ir\_transition

```
prepare_csv_semantic_ir_transition(directory: 'TDSDirectory', csv_id: 'str', source_candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'CSVSemanticIRTransition | Mapping[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Create one authorized in-memory lifecycle transition without writing TDS artifacts.

### validate\_csv\_semantic\_ir\_lifecycle

```
validate_csv_semantic_ir_lifecycle(lifecycle: 'CSVSemanticIRLifecycle | Mapping[str, Any]', *, source_candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'CSVSemanticIRValidationReport | Mapping[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### replay\_csv\_semantic\_ir\_lifecycle

```
replay_csv_semantic_ir_lifecycle(directory: 'TDSDirectory', csv_id: 'str', source_candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'CSVSemanticIRCandidate | Mapping[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Recompute the current result and compare it with the supplied source object to detect drift or tampering.

### csv\_semantic\_ir\_transition\_authorization\_fingerprint

```
csv_semantic_ir_transition_authorization_fingerprint(authorization: 'CSVSemanticIRTransitionAuthorization | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### csv\_semantic\_ir\_transition\_fingerprint

```
csv_semantic_ir_transition_fingerprint(record: 'CSVSemanticIRTransitionRecord | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### csv\_semantic\_ir\_lifecycle\_fingerprint

```
csv_semantic_ir_lifecycle_fingerprint(lifecycle: 'CSVSemanticIRLifecycle | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### csv\_semantic\_ir\_lifecycle\_summary

```
csv_semantic_ir_lifecycle_summary(lifecycle: 'CSVSemanticIRLifecycle') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### csv\_semantic\_ir\_lifecycle\_replay\_summary

```
csv_semantic_ir_lifecycle_replay_summary(report: 'CSVSemanticIRLifecycleReplayReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### prepare\_csv\_semantic\_ir\_transition\_batch

```
prepare_csv_semantic_ir_transition_batch(directory: 'TSDDirectory', csv_id: 'str', source_candidate: 'CSVSemanticIRCandidate')
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Preflight and simulate an all-or-nothing batch of authorized lifecycle transitions.

### validate\_csv\_semantic\_ir\_transition\_batch

```
validate_csv_semantic_ir_transition_batch(receipt: 'CSVSemanticIRBatchReceipt | Mapping[str, Any]', *, source_candidate: 'CSVSemanticIRCandidate')
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### replay\_csv\_semantic\_ir\_transition\_batch

```
replay_csv_semantic_ir_transition_batch(directory: 'TSDDirectory', csv_id: 'str', source_candidate: 'CSVSemanticIRCandidate')
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Recompute the current result and compare it with the supplied source object to detect drift or tampering.

### csv\_semantic\_ir\_transition\_request\_fingerprint

```
csv_semantic_ir_transition_request_fingerprint(request: 'CSVSemanticIRTransitionRequest | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### csv\_semantic\_ir\_batch\_authorization\_fingerprint

```
csv_semantic_ir_batch_authorization_fingerprint(authorization: 'CSVSemanticIRBatchAuthorization | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### csv\_semantic\_ir\_transition\_batch\_fingerprint

```
csv_semantic_ir_transition_batch_fingerprint(batch: 'CSVSemanticIRTransitionBatch | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### csv\_semantic\_ir\_batch\_receipt\_fingerprint

```
csv_semantic_ir_batch_receipt_fingerprint(receipt: 'CSVSemanticIRBatchReceipt | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### csv\_semantic\_ir\_transition\_batch\_summary

```
csv_semantic_ir_transition_batch_summary(receipt: 'CSVSemanticIRBatchReceipt') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### csv\_semantic\_ir\_transition\_batch\_replay\_summary

```
csv_semantic_ir_transition_batch_replay_summary(report: 'CSVSemanticIRBatchReplayReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

---

#### **v3.5.2 state boundary**

Admitted states are proposed, validated, and contested. The contract does not admit committed or superseded, and it performs no automatic transition.

## 14. CSV operational call index

This section lists every public function exported by `staqtapp_tds.csv_layer`, grouped by implementation module. The earlier sections explain the primary workflow; this index is for exact lookup.

### 14.1 browser\_monitor

#### prepare\_csv\_interpole\_browser\_monitor\_snapshot

```
prepare_csv_interpole_browser_monitor_snapshot(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7
```

**Import:** `staqtapp_tds.csv_layer.browser_monitor`

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

#### csv\_interpole\_browser\_monitor\_display\_contract

```
csv_interpole_browser_monitor_display_contract() -> 'dict[str, Any]'
```

**Import:** `staqtapp_tds.csv_layer.browser_monitor`

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### csv\_interpole\_browser\_monitor\_display\_contract\_fingerprint

```
csv_interpole_browser_monitor_display_contract_fingerprint() -> 'str'
```

**Import:** `staqtapp_tds.csv_layer.browser_monitor`

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

#### csv\_interpole\_browser\_monitor\_replay\_summary

```
csv_interpole_browser_monitor_replay_summary(report: 'CSVInterpoleMonitorReplayReport') -> 'dict[str, Any]'
```

**Import:** `staqtapp_tds.csv_layer.browser_monitor`

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

#### csv\_interpole\_browser\_monitor\_snapshot\_fingerprint

```
csv_interpole_browser_monitor_snapshot_fingerprint(snapshot: 'CSVInterpoleBrowserMonitorSnapshot | Mapping[str, Any]') -
```

**Import:** `staqtapp_tds.csv_layer.browser_monitor`

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

#### csv\_interpole\_browser\_monitor\_summary

```
csv_interpole_browser_monitor_summary(snapshot: 'CSVInterpoleBrowserMonitorSnapshot') -> 'dict[str, Any]'
```

**Import:** `staqtapp_tds.csv_layer.browser_monitor`

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

#### csv\_interpole\_monitor\_delivery\_manifest

```
csv_interpole_monitor_delivery_manifest(*, release_version: 'str' = '3.4.10', package_name: 'str' = 'staqtapp_tds_v3.4.1
```

**Import:** `staqtapp_tds.csv_layer.browser_monitor`

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### csv\_interpole\_monitor\_icon\_registry

```
csv_interpole_monitor_icon_registry() -> 'dict[str, str]'
```

**Import:** `staqtapp_tds.csv_layer.browser_monitor`

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### replay\_csv\_interpole\_browser\_monitor\_snapshot

```
replay_csv_interpole_browser_monitor_snapshot(directory: 'TSDirectory', csv_id: 'str', source_snapshot: 'CSVInterpoleBr
```

**Import:** `staqtapp_tds.csv_layer.browser_monitor`

**Behavior:** Recompute the current result and compare it with the supplied source object to detect drift or tampering.

#### validate\_csv\_interpole\_browser\_monitor\_snapshot

```
validate_csv_interpole_browser_monitor_snapshot(snapshot: 'CSVInterpoleBrowserMonitorSnapshot | Mapping[str, Any]', *, p
```

**Import:** `staqtapp_tds.csv_layer.browser_monitor`

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API

contract.

### validate\_csv\_interpolate\_monitor\_icon\_registry

```
validate_csv_interpolate_monitor_icon_registry(registry: 'Mapping[str, str] | None' = None, *, svg_payloads: 'Mapping[str,
```

**Import:** staqtapp\_tds.csv\_layer.browser\_monitor

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## 14.2 dialect

### detect\_csv\_dialect

```
detect_csv_dialect(text: 'str', *, delimiters: 'Iterable[str]' = (';', '\t', '|')) -> 'CSVDialectFingerprint'
```

**Import:** staqtapp\_tds.csv\_layer.dialect

**Behavior:** Detect the requested condition or format and return deterministic evidence.

### dialect\_to\_csv\_kwargs

```
dialect_to_csv_kwargs(dialect: 'CSVDialectFingerprint') -> 'dict[str, object]'
```

**Import:** staqtapp\_tds.csv\_layer.dialect

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## 14.3 exporter

### export\_canonical\_csv

```
export_canonical_csv(directory: 'TSDirectory', csv_id: 'str', *, lineterminator: 'str' = '\n') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.exporter

**Behavior:** Return a deterministic canonical CSV rendering while leaving the preserved source unchanged.

### export\_original\_csv

```
export_original_csv(directory: 'TSDirectory', csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.exporter

**Behavior:** Return the preserved original CSV text without canonical rewriting.

### load\_csv\_artifact

```
load_csv_artifact(directory: 'TSDirectory', csv_id: 'str', artifact: 'str') -> 'Any'
```

**Import:** staqtapp\_tds.csv\_layer.exporter

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### prove\_original\_roundtrip

```
prove_original_roundtrip(directory: 'TSDirectory', csv_id: 'str', *, overwrite: 'bool' = True) -> 'CSVRoundTripReport'
```

**Import:** staqtapp\_tds.csv\_layer.exporter

**Behavior:** Write and return original-byte round-trip proof.

## 14.4 importer

### import\_csv\_bytes

```
import_csv_bytes(directory: 'TSDirectory', raw: 'bytes', *, source_name: 'str' = 'csv_source.csv', encoding: 'str' = 'u
```

**Import:** staqtapp\_tds.csv\_layer.importer

**Behavior:** Import immutable CSV bytes and write the core CSV artifact family into a TDS directory.

### import\_csv\_file

```
import_csv_file(directory: 'TSDirectory', path: 'str | Path', *, encoding: 'str' = 'utf-8', csv_id: 'str | None' = None
```

**Import:** staqtapp\_tds.csv\_layer.importer

**Behavior:** Read a file, preserve its bytes/text identity, and write the core CSV artifact family.

### load\_csv\_manifest

```
load_csv_manifest(directory: 'TSDirectory', csv_id: 'str') -> 'CSVImportManifest'
```

**Import:** staqtapp\_tds.csv\_layer.importer

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

## 14.5 interpolate

### prepare\_csv\_interpole\_timeline

```
prepare_csv_interpole_timeline(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: 'str' = 'utf-8')
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### commit\_csv\_interpole\_timeline\_report

```
commit_csv_interpole_timeline_report(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, overwrite: bool = False)
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### load\_csv\_interpole\_timeline\_report

```
load_csv_interpole_timeline_report(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVInterpoleTimelineReport'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_interpole\_timeline

```
validate_csv_interpole_timeline(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: 'str' = 'utf-8')
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### prepare\_csv\_interpole\_determinant\_vector

```
prepare_csv_interpole_determinant_vector(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: 'str' = 'utf-8')
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### commit\_csv\_interpole\_determinant\_vector\_report

```
commit_csv_interpole_determinant_vector_report(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, overwrite: bool = False)
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### load\_csv\_interpole\_determinant\_vector\_report

```
load_csv_interpole_determinant_vector_report(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVInterpoleDeterminantVectorReport'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_interpole\_determinant\_vector

```
validate_csv_interpole_determinant_vector(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: 'str' = 'utf-8')
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### prepare\_csv\_interpole\_timeline\_ring

```
prepare_csv_interpole_timeline_ring(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: 'str' = 'utf-8')
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### commit\_csv\_interpole\_timeline\_ring\_report

```
commit_csv_interpole_timeline_ring_report(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, overwrite: bool = False)
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### load\_csv\_interpole\_timeline\_ring\_report

```
load_csv_interpole_timeline_ring_report(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVInterpoleTimelineRingReport'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_interpole\_timeline\_ring

```
validate_csv_interpole_timeline_ring(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoded: bool = False) -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### csv\_interpole\_timeline\_report\_key

```
csv_interpole_timeline_report_key(csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### csv\_interpole\_timeline\_summary

```
csv_interpole_timeline_summary(report: 'CSVInterpoleTimelineReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### csv\_interpole\_determinant\_vector\_report\_key

```
csv_interpole_determinant_vector_report_key(csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### csv\_interpole\_determinant\_vector\_summary

```
csv_interpole_determinant_vector_summary(report: 'CSVInterpoleDeterminantVectorReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### csv\_interpole\_timeline\_ring\_report\_key

```
csv_interpole_timeline_ring_report_key(csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### csv\_interpole\_timeline\_ring\_summary

```
csv_interpole_timeline_ring_summary(report: 'CSVInterpoleTimelineRingReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.interpole

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

## 14.6 kernel

### prepare\_csv\_kernel\_readiness\_contract

```
prepare_csv_kernel_readiness_contract(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7) -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.kernel

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### commit\_csv\_kernel\_readiness\_contract\_report

```
commit_csv_kernel_readiness_contract_report(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7, encoded: bool = False) -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.kernel

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### load\_csv\_kernel\_readiness\_contract\_report

```
load_csv_kernel_readiness_contract_report(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVKernelReadinessReport'
```

**Import:** staqtapp\_tds.csv\_layer.kernel

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_kernel\_readiness\_contract

```
validate_csv_kernel_readiness_contract(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7) -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.kernel

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### csv\_kernel\_readiness\_report\_key

```
csv_kernel_readiness_report_key(csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.kernel

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### csv\_kernel\_readiness\_contract\_summary

```
csv_kernel_readiness_contract_summary(report: 'CSVKernelReadinessReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.kernel

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

## 14.7 manifest

### artifact\_keys

```
artifact_keys(csv_id: 'str') -> 'dict[str, str]'
```

**Import:** staqtapp\_tds.csv\_layer.manifest

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### build\_manifest

```
build_manifest(*, source_name: 'str', text: 'str', raw: 'bytes', encoding: 'str', dialect: 'CSVDialectFingerprint', csv_
```

**Import:** staqtapp\_tds.csv\_layer.manifest

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### iter\_rows

```
iter_rows(text: 'str', dialect: 'CSVDialectFingerprint') -> 'Iterator[list[str]]'
```

**Import:** staqtapp\_tds.csv\_layer.manifest

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### row\_count\_and\_column\_count

```
row_count_and_column_count(text: 'str', dialect: 'CSVDialectFingerprint') -> 'tuple[int, int]'
```

**Import:** staqtapp\_tds.csv\_layer.manifest

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### safe\_csv\_id

```
safe_csv_id(source_name: 'str', raw: 'bytes') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.manifest

**Behavior:** Validate or normalize an identifier using the public safety contract.

### validate\_csv\_id

```
validate_csv_id(csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.manifest

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### is\_safe\_csv\_id

```
is_safe_csv_id(csv_id: 'str') -> 'bool'
```

**Import:** staqtapp\_tds.csv\_layer.manifest

**Behavior:** Validate or normalize an identifier using the public safety contract.

### sha256\_hex

```
sha256_hex(data: 'bytes') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.manifest

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## 14.8 native\_row\_anchor

### prepare\_csv\_native\_row\_anchor\_kernel

```
prepare_csv_native_row_anchor_kernel(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7, use_nati
```

**Import:** staqtapp\_tds.csv\_layer.native\_row\_anchor

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation

states otherwise.

### load\_csv\_native\_row\_anchor\_kernel\_report

```
load_csv_native_row_anchor_kernel_report(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVNativeRowAnchorKernelReport'
```

**Import:** staqtapp\_tds.csv\_layer.native\_row\_anchor

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_native\_row\_anchor\_kernel

```
validate_csv_native_row_anchor_kernel(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7) -> 'CSVNativeRowAnchorKernelReport'
```

**Import:** staqtapp\_tds.csv\_layer.native\_row\_anchor

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### commit\_csv\_native\_row\_anchor\_kernel\_report

```
commit_csv_native_row_anchor_kernel_report(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7, use_)
```

**Import:** staqtapp\_tds.csv\_layer.native\_row\_anchor

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### csv\_native\_row\_anchor\_kernel\_report\_key

```
csv_native_row_anchor_kernel_report_key(csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.native\_row\_anchor

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### csv\_native\_row\_anchor\_kernel\_summary

```
csv_native_row_anchor_kernel_summary(report: 'CSVNativeRowAnchorKernelReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.native\_row\_anchor

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

## 14.9 native\_scan

### prepare\_csv\_native\_scan\_kernel\_prototype

```
prepare_csv_native_scan_kernel_prototype(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7, use_)
```

**Import:** staqtapp\_tds.csv\_layer.native\_scan

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### load\_csv\_native\_scan\_kernel\_prototype\_report

```
load_csv_native_scan_kernel_prototype_report(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVNativeScanKernelReport'
```

**Import:** staqtapp\_tds.csv\_layer.native\_scan

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_native\_scan\_kernel\_prototype

```
validate_csv_native_scan_kernel_prototype(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7) -> 'CSVNativeScanKernelReport'
```

**Import:** staqtapp\_tds.csv\_layer.native\_scan

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### commit\_csv\_native\_scan\_kernel\_prototype\_report

```
commit_csv_native_scan_kernel_prototype_report(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7, use_)
```

**Import:** staqtapp\_tds.csv\_layer.native\_scan

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### csv\_native\_scan\_kernel\_report\_key

```
csv_native_scan_kernel_report_key(csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.native\_scan

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### csv\_native\_scan\_kernel\_summary

```
csv_native_scan_kernel_summary(report: 'CSVNativeScanKernelReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.native\_scan

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

## 14.10 performance\_gates

### prepare\_csv\_kernel\_performance\_gates

```
prepare_csv_kernel_performance_gates(directory: 'TDSDirectory', csv_id: 'str', *, max_report_json_bytes: 'int' = 65536,
```

**Import:** staqtapp\_tds.csv\_layer.performance\_gates

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### load\_csv\_kernel\_performance\_gate\_report

```
load_csv_kernel_performance_gate_report(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVKernelPerformanceGateReport'
```

**Import:** staqtapp\_tds.csv\_layer.performance\_gates

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### validate\_csv\_kernel\_performance\_gate\_report

```
validate_csv_kernel_performance_gate_report(directory: 'TDSDirectory', csv_id: 'str', *, max_report_json_bytes: 'int' | N
```

**Import:** staqtapp\_tds.csv\_layer.performance\_gates

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### commit\_csv\_kernel\_performance\_gate\_report

```
commit_csv_kernel_performance_gate_report(directory: 'TDSDirectory', csv_id: 'str', *, max_report_json_bytes: 'int' = 65
```

**Import:** staqtapp\_tds.csv\_layer.performance\_gates

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### csv\_kernel\_performance\_gate\_report\_key

```
csv_kernel_performance_gate_report_key(csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.performance\_gates

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### csv\_kernel\_performance\_gate\_summary

```
csv_kernel_performance_gate_summary(report: 'CSVKernelPerformanceGateReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.performance\_gates

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

## 14.11 reload

### reload\_csv\_artifacts

```
reload_csv_artifacts(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVReloadedArtifacts'
```

**Import:** staqtapp\_tds.csv\_layer.reload

**Behavior:** Reload the committed CSV artifact family into typed value objects.

## 14.12 row\_offsets

### build\_row\_offset\_map

```
build_row_offset_map(text: 'str', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8', raw: 'bytes | None' =
```

**Import:** staqtapp\_tds.csv\_layer.row\_offsets

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### logical\_record\_offsets

```
logical_record_offsets(text: 'str', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8') -> 'tuple[int, ...]'
```

**Import:** staqtapp\_tds.csv\_layer.row\_offsets

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### logical\_record\_offsets\_bytes

```
logical_record_offsets_bytes(raw: 'bytes | bytearray | memoryview', dialect: 'CSVDialectFingerprint', *, encoding: 'str'
```

**Import:** staqtapp\_tds.csv\_layer.row\_offsets

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### pack\_csv\_row\_offsets

```
pack_csv_row_offsets(row_offsets: 'Iterable[int]') -> 'bytes'
```

**Import:** staqtapp\_tds.csv\_layer.row\_offsets

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### unpack\_csv\_row\_offsets

```
unpack_csv_row_offsets(payload: 'bytes | bytearray | memoryview') -> 'tuple[int, ...]'
```

**Import:** staqtapp\_tds.csv\_layer.row\_offsets

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## 14.13 scan\_artifacts

### csv\_scan\_artifact\_keys

```
csv_scan_artifact_keys(csv_id: 'str') -> 'dict[str, str]'
```

**Import:** staqtapp\_tds.csv\_layer.scan\_artifacts

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### load\_csv\_row\_anchor\_profile

```
load_csv_row_anchor_profile(directory: 'TSDirectory', csv_id: 'str') -> 'CSVRowAnchorProfile'
```

**Import:** staqtapp\_tds.csv\_layer.scan\_artifacts

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### load\_csv\_scan\_materialization\_report

```
load_csv_scan_materialization_report(directory: 'TSDirectory', csv_id: 'str') -> 'CSVScanArtifactReport'
```

**Import:** staqtapp\_tds.csv\_layer.scan\_artifacts

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### load\_csv\_scan\_profile

```
load_csv_scan_profile(directory: 'TSDirectory', csv_id: 'str') -> 'CSVScanProfile'
```

**Import:** staqtapp\_tds.csv\_layer.scan\_artifacts

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### materialize\_csv\_scan\_artifacts

```
materialize_csv_scan_artifacts(directory: 'TSDirectory', csv_id: 'str', *, include_row_anchors: 'bool' = True, chunk_size: 'int' = 1024)
```

**Import:** staqtapp\_tds.csv\_layer.scan\_artifacts

**Behavior:** Persist scan profile and row-anchor evidence as bounded derived artifacts.

### validate\_materialized\_csv\_scan\_artifacts

```
validate_materialized_csv_scan_artifacts(directory: 'TSDirectory', csv_id: 'str', *, require_row_anchors: 'bool' = True)
```

**Import:** staqtapp\_tds.csv\_layer.scan\_artifacts

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## 14.14 scanner

### scan\_csv\_bytes

```
scan_csv_bytes(raw: 'BytesLike', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8', chunk_size: 'int | None' = None)
```

**Import:** staqtapp\_tds.csv\_layer.scanner

**Behavior:** Build a deterministic reference scan profile from immutable CSV bytes.

### scan\_csv\_text

```
scan_csv_text(text: 'str', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8', chunk_size: 'int | None' = None)
```

**Import:** staqtapp\_tds.csv\_layer.scanner

**Behavior:** Scan immutable input and return a deterministic profile or ranking result.

### validate\_csv\_row\_anchors

```
validate_csv_row_anchors(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVRowAnchorProfile'
```

**Import:** staqtapp\_tds.csv\_layer.scanner

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

contract.

### scan\_csv\_text\_row\_anchors

```
scan_csv_text_row_anchors(text: 'str', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8', chunk_size: 'int
```

**Import:** staqtapp\_tds.csv\_layer.scanner

**Behavior:** Scan immutable input and return a deterministic profile or ranking result.

### scan\_csv\_row\_anchors

```
scan_csv_row_anchors(raw: 'BytesLike', dialect: 'CSVDialectFingerprint', *, encoding: 'str' = 'utf-8', chunk_size: 'int
```

**Import:** staqtapp\_tds.csv\_layer.scanner

**Behavior:** Build row-anchor hashes and profile information from immutable CSV bytes.

### validate\_csv\_scan\_profile

```
validate_csv_scan_profile(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVScanParit
```

**Import:** staqtapp\_tds.csv\_layer.scanner

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## 14.15 security

### validate\_csv\_artifact\_security

```
validate_csv_artifact_security(directory: 'TDSDirectory', csv_id: 'str', *, include_scan_artifacts: 'bool' = False) -> '
```

**Import:** staqtapp\_tds.csv\_layer.security

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### validate\_csv\_artifact\_key

```
validate_csv_artifact_key(key: 'str', csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.security

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## 14.16 semantic\_handoff

### prepare\_csv\_semantic\_ir\_handoff

```
prepare_csv_semantic_ir_handoff(directory: 'TDSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = 7, source_monito
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_handoff

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### validate\_csv\_semantic\_ir\_handoff

```
validate_csv_semantic_ir_handoff(report: 'CSVSemanticIRHandoffReport | Mapping[str, Any]') -> 'CSVSemanticIRHandoffValid
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_handoff

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### csv\_semantic\_ir\_handoff\_fingerprint

```
csv_semantic_ir_handoff_fingerprint(report: 'CSVSemanticIRHandoffReport | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_handoff

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### csv\_semantic\_ir\_handoff\_summary

```
csv_semantic_ir_handoff_summary(report: 'CSVSemanticIRHandoffReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_handoff

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

## 14.17 semantic\_ir

### csv\_semantic\_ir\_declaration\_fingerprint

```
csv_semantic_ir_declaration_fingerprint(declaration: 'CSVSemanticIRDeclaration | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### prepare\_csv\_semantic\_ir\_candidate

```
prepare_csv_semantic_ir_candidate(directory: 'TSDirectory', csv_id: 'str', declarations: 'Sequence[CSVSemanticIRDeclaration]') -> 'CSVSemanticIRCandidate'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Create an in-memory, caller-declared Semantic IR candidate after explicit opt-in and evidence validation.

### validate\_csv\_semantic\_ir\_candidate

```
validate_csv_semantic_ir_candidate(candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'CSVSemanticIRValidationResult'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### csv\_semantic\_ir\_candidate\_fingerprint

```
csv_semantic_ir_candidate_fingerprint(candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### replay\_csv\_semantic\_ir\_candidate

```
replay_csv_semantic_ir_candidate(directory: 'TSDirectory', csv_id: 'str', source_candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'CSVSemanticIRCandidate'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Recompute the current result and compare it with the supplied source object to detect drift or tampering.

### csv\_semantic\_ir\_candidate\_summary

```
csv_semantic_ir_candidate_summary(candidate: 'CSVSemanticIRCandidate') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### csv\_semantic\_ir\_replay\_summary

```
csv_semantic_ir_replay_summary(report: 'CSVSemanticIRReplayReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

## 14.18 semantic\_ir\_lifecycle

### csv\_semantic\_ir\_transition\_authorization\_fingerprint

```
csv_semantic_ir_transition_authorization_fingerprint(authorization: 'CSVSemanticIRTransitionAuthorization | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### csv\_semantic\_ir\_transition\_fingerprint

```
csv_semantic_ir_transition_fingerprint(record: 'CSVSemanticIRTransitionRecord | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

### prepare\_csv\_semantic\_ir\_transition

```
prepare_csv_semantic_ir_transition(directory: 'TSDirectory', csv_id: 'str', source_candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'CSVSemanticIRTransitionRecord'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Create one authorized in-memory lifecycle transition without writing TDS artifacts.

### validate\_csv\_semantic\_ir\_lifecycle

```
validate_csv_semantic_ir_lifecycle(lifecycle: 'CSVSemanticIRLifecycle | Mapping[str, Any]', *, source_candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'bool'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### csv\_semantic\_ir\_lifecycle\_fingerprint

```
csv_semantic_ir_lifecycle_fingerprint(lifecycle: 'CSVSemanticIRLifecycle | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

## replay\_csv\_semantic\_ir\_lifecycle

```
replay_csv_semantic_ir_lifecycle(directory: 'TDSDirectory', csv_id: 'str', source_candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Recompute the current result and compare it with the supplied source object to detect drift or tampering.

## csv\_semantic\_ir\_lifecycle\_summary

```
csv_semantic_ir_lifecycle_summary(lifecycle: 'CSVSemanticIRLifecycle') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

## csv\_semantic\_ir\_lifecycle\_replay\_summary

```
csv_semantic_ir_lifecycle_replay_summary(report: 'CSVSemanticIRLifecycleReplayReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

# 14.19 semantic\_ir\_lifecycle\_batch

## csv\_semantic\_ir\_transition\_request\_fingerprint

```
csv_semantic_ir_transition_request_fingerprint(request: 'CSVSemanticIRTransitionRequest | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

## csv\_semantic\_ir\_batch\_authorization\_fingerprint

```
csv_semantic_ir_batch_authorization_fingerprint(authorization: 'CSVSemanticIRBatchAuthorization | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

## csv\_semantic\_ir\_transition\_batch\_fingerprint

```
csv_semantic_ir_transition_batch_fingerprint(batch: 'CSVSemanticIRTransitionBatch | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

## csv\_semantic\_ir\_batch\_receipt\_fingerprint

```
csv_semantic_ir_batch_receipt_fingerprint(receipt: 'CSVSemanticIRBatchReceipt | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Return the deterministic fingerprint for the supplied contract object.

## prepare\_csv\_semantic\_ir\_transition\_batch

```
prepare_csv_semantic_ir_transition_batch(directory: 'TDSDirectory', csv_id: 'str', source_candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Preflight and simulate an all-or-nothing batch of authorized lifecycle transitions.

## validate\_csv\_semantic\_ir\_transition\_batch

```
validate_csv_semantic_ir_transition_batch(receipt: 'CSVSemanticIRBatchReceipt | Mapping[str, Any]', *, source_candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## replay\_csv\_semantic\_ir\_transition\_batch

```
replay_csv_semantic_ir_transition_batch(directory: 'TDSDirectory', csv_id: 'str', source_candidate: 'CSVSemanticIRCandidate | Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Recompute the current result and compare it with the supplied source object to detect drift or tampering.

## csv\_semantic\_ir\_transition\_batch\_summary

```
csv_semantic_ir_transition_batch_summary(receipt: 'CSVSemanticIRBatchReceipt') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

## csv\_semantic\_ir\_transition\_batch\_replay\_summary

```
csv_semantic_ir_transition_batch_replay_summary(report: 'CSVSemanticIRBatchReplayReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.semantic\_ir\_lifecycle\_batch

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

## 14.20 storage\_adapter

### validate\_csv\_storage\_bridge\_commit

```
validate_csv_storage_bridge_commit(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVStorageBridgeCommitReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### validate\_csv\_storage\_adapter\_binding

```
validate_csv_storage_adapter_binding(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVStorageAdapterBindingReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### validate\_csv\_storage\_adapter\_replay

```
validate_csv_storage_adapter_replay(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVStorageAdapterReplayReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### validate\_csv\_native\_storage\_commit

```
validate_csv_native_storage_commit(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: 'str' = 'utf-8') -> 'CSVNativeStorageCommitReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### validate\_csv\_native\_storage\_revalidation

```
validate_csv_native_storage_revalidation(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: 'str' = 'utf-8') -> 'CSVNativeStorageRevalidationReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### prepare\_csv\_storage\_bridge\_commit

```
prepare_csv_storage_bridge_commit(directory: 'TSDirectory', csv_id: 'str', *, include_scan_artifacts: 'bool' = False, chunk_size: 'int | None' = None) -> 'CSVStorageBridgeCommitReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### prepare\_csv\_storage\_adapter\_binding

```
prepare_csv_storage_adapter_binding(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVStorageAdapterBindingReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### prepare\_csv\_storage\_adapter\_replay

```
prepare_csv_storage_adapter_replay(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None) -> 'CSVStorageAdapterReplayReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### prepare\_csv\_native\_storage\_revalidation

```
prepare_csv_native_storage_revalidation(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, encoding: 'str' = 'utf-8') -> 'CSVNativeStorageRevalidationReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Build a deterministic read-only report or candidate from current evidence; does not publish the result unless the specific implementation states otherwise.

### load\_csv\_storage\_bridge\_commit\_report

```
load_csv_storage_bridge_commit_report(directory: 'TSDirectory', csv_id: 'str') -> 'CSVStorageBridgeCommitReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### load\_csv\_storage\_adapter\_replay\_report

```
load_csv_storage_adapter_replay_report(directory: 'TSDDirectory', csv_id: 'str') -> 'CSVStorageAdapterReplayReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### load\_csv\_native\_storage\_commit\_report

```
load_csv_native_storage_commit_report(directory: 'TSDDirectory', csv_id: 'str') -> 'CSVNativeStorageCommitReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### load\_csv\_native\_storage\_revalidation\_report

```
load_csv_native_storage_revalidation_report(directory: 'TSDDirectory', csv_id: 'str') -> 'CSVNativeStorageRevalidationReport'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### csv\_storage\_bridge\_commit\_summary

```
csv_storage_bridge_commit_summary(report: 'CSVStorageBridgeCommitReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### csv\_storage\_adapter\_binding\_summary

```
csv_storage_adapter_binding_summary(report: 'CSVStorageAdapterBindingReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### csv\_storage\_adapter\_replay\_summary

```
csv_storage_adapter_replay_summary(report: 'CSVStorageAdapterReplayReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### csv\_native\_storage\_commit\_summary

```
csv_native_storage_commit_summary(report: 'CSVNativeStorageCommitReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### csv\_native\_storage\_revalidation\_summary

```
csv_native_storage_revalidation_summary(report: 'CSVNativeStorageRevalidationReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### csv\_storage\_bridge\_commit\_report\_key

```
csv_storage_bridge_commit_report_key(csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### csv\_storage\_adapter\_replay\_report\_key

```
csv_storage_adapter_replay_report_key(csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### csv\_native\_storage\_commit\_report\_key

```
csv_native_storage_commit_report_key(csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### csv\_native\_storage\_revalidation\_report\_key

```
csv_native_storage_revalidation_report_key(csv_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### commit\_csv\_storage\_bridge\_manifest

```
commit_csv_storage_bridge_manifest(directory: 'TSDirectory', csv_id: 'str', *, include_scan_artifacts: 'bool' = False,
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### commit\_csv\_storage\_adapter\_replay\_report

```
commit_csv_storage_adapter_replay_report(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, o
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### commit\_csv\_native\_storage\_artifacts

```
commit_csv_native_storage_artifacts(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = None, overwr
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

### commit\_csv\_native\_storage\_revalidation\_report

```
commit_csv_native_storage_revalidation_report(directory: 'TSDirectory', csv_id: 'str', *, chunk_size: 'int | None' = No
```

**Import:** staqtapp\_tds.csv\_layer.storage\_adapter

**Behavior:** Validate current prerequisites and persist the resulting report/artifact using the TDS artifact contract.

## 14.21 storage\_bridge

### csv\_storage\_bridge\_plan

```
csv_storage_bridge_plan(csv_id: 'str', *, include_scan_artifacts: 'bool' = False, include_transaction_report: 'bool' = F
```

**Import:** staqtapp\_tds.csv\_layer.storage\_bridge

**Behavior:** Return the deterministic operation plan without performing storage writes.

### csv\_storage\_bridge\_preflight\_summary

```
csv_storage_bridge_preflight_summary(report: 'CSVStorageBridgePreflightReport') -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.csv\_layer.storage\_bridge

**Behavior:** Return a compact JSON-safe summary suitable for logs, Browser payloads, or external reporting.

### validate\_csv\_storage\_bridge\_preflight

```
validate_csv_storage_bridge_preflight(directory: 'TSDirectory', csv_id: 'str', *, include_scan_artifacts: 'bool' = Fals
```

**Import:** staqtapp\_tds.csv\_layer.storage\_bridge

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## 14.22 transaction

### begin\_csv\_artifact\_transaction

```
begin_csv_artifact_transaction(directory: 'TSDirectory', raw: 'bytes', *, source_name: 'str' = 'csv_source.csv', encodi
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Stage a complete core CSV artifact family under a transaction namespace.

### commit\_csv\_artifact\_transaction

```
commit_csv_artifact_transaction(directory: 'TSDirectory', csv_id: 'str', transaction_id: 'str', *, overwrite: 'bool' =
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Validate a staged CSV transaction and publish the complete artifact family.

### csv\_artifact\_transaction\_keys

```
csv_artifact_transaction_keys(csv_id: 'str', transaction_id: 'str') -> 'dict[str, str]'
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Return deterministic TDS artifact key name(s) for the requested object.

### detect\_partial\_csv\_artifacts

```
detect_partial_csv_artifacts(directory: 'TSDirectory', csv_id: 'str') -> 'CSVArtifactTransactionReport'
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Detect the requested condition or format and return deterministic evidence.

### load\_csv\_artifact\_transaction\_report

```
load_csv_artifact_transaction_report(directory: 'TDSDirectory', csv_id: 'str', transaction_id: 'str | None' = None) -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### new\_csv\_transaction\_id

```
new_csv_transaction_id() -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### recover\_csv\_artifact\_transaction

```
recover_csv_artifact_transaction(directory: 'TDSDirectory', csv_id: 'str', transaction_id: 'str', *, commit_staged: 'bool')
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Inspect or recover an interrupted CSV artifact transaction.

### validate\_csv\_artifact\_transaction

```
validate_csv_artifact_transaction(directory: 'TDSDirectory', csv_id: 'str', transaction_id: 'str') -> 'CSVArtifactTransactionReport'
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### validate\_csv\_transaction\_id

```
validate_csv_transaction_id(transaction_id: 'str') -> 'str'
```

**Import:** staqtapp\_tds.csv\_layer.transaction

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## 14.23 validator

### validate\_csv\_artifacts

```
validate_csv_artifacts(directory: 'TDSDirectory', csv_id: 'str') -> 'CSVArtifactValidationReport'
```

**Import:** staqtapp\_tds.csv\_layer.validator

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## 15. Driver TDDL, bytecode, Foundry, and Runtime Manager

The Driver subsystem uses a closed deterministic grammar and bytecode package. Driver Foundry can validate, compile, audit, test, and submit candidate evidence. It cannot sign or activate drivers. Execution authority remains in the Runtime Manager and Registry policy chain.

```
from staqtapp_tds.drivers import DriverFoundry, DriverVMRuntime

source = """
driver SearchPolicies v1

manifest:
  kind = "search"
  description = "Find policy records"
  safety = "bounded"

requires:
  capability registry.scan
  capability manifest.read

limits:
  max_scan = 1000
  max_depth = 4
  timeout_ms = 250

program:
  SCAN scope=".tds" recursive=true limit=1000 depth=4
  READ target="manifest"
  MATCH field="manifest.kind" eq="policy"
  EXTRACT from="manifest" fields=["policy_id", "version"]
  EMIT mode="list" limit=10
  HALT

evolution:
  deny external_io
  max_delta = 1
"""

foundry = DriverFoundry()
built = foundry.compile_driver(source)
if built.ok:
    vm = DriverVMRuntime(max_cost=100_000)
    vm.load(built.package)
    records = [
        {"manifest": {"kind": "policy", "policy_id": "p1", "version": 1}}
    ]
    result = vm.execute({"records": records})
```

### parse\_tddl

```
parse_tddl(source: 'str') -> 'TDDLProgram'
```

**Import:** staqtapp\_tds.drivers.tddl

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### validate\_tddl

```
validate_tddl(program: 'TDDLProgram') -> 'None'
```

**Import:** staqtapp\_tds.drivers.tddl

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### instruction\_specs

```
instruction_specs() -> 'Mapping[str, InstructionSpec]'
```

**Import:** staqtapp\_tds.drivers.tddl

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## compile\_tddl

```
compile_tddl(source_or_program: 'str | TDDLProgram') -> 'BytecodePackage'
```

**Import:** staqtapp\_tds.drivers.bytecode

**Behavior:** Validate TDDL source and compile it into a deterministic bytecode package.

## compile\_program

```
compile_program(program: 'TDDLProgram', *, source_material: 'str | Mapping[str, Any] | None' = None) -> 'BytecodePackage'
```

**Import:** staqtapp\_tds.drivers.bytecode

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## validate\_bytecode\_package

```
validate_bytecode_package(package: 'BytecodePackage') -> 'None'
```

**Import:** staqtapp\_tds.drivers.bytecode

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## decompile\_to\_ir

```
decompile_to_ir(package: 'BytecodePackage') -> 'TDDLProgram'
```

**Import:** staqtapp\_tds.drivers.bytecode

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## opcode\_table

```
opcode_table() -> 'Mapping[str, Mapping[str, Any]]'
```

**Import:** staqtapp\_tds.drivers.bytecode

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## audit\_vm\_contract

```
audit_vm_contract(package: 'BytecodePackage') -> 'None'
```

**Import:** staqtapp\_tds.drivers.audit

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## vm\_contract\_table

```
vm_contract_table() -> 'Mapping[str, Mapping[str, Any]]'
```

**Import:** staqtapp\_tds.drivers.audit

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## sign\_payload

```
sign_payload(payload: 'bytes', *, signer: 'str', secret: 'bytes') -> 'str'
```

**Import:** staqtapp\_tds.drivers.signature

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## verify\_signature

```
verify_signature(payload: 'bytes', signature: 'str', *, signer: 'str', secret: 'bytes') -> 'bool'
```

**Import:** staqtapp\_tds.drivers.signature

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## DriverFoundry

```
DriverFoundry(*, policy: 'DriverFoundryPolicy | None' = None) -> 'None'
```

### DriverFoundry.capability\_matrix

```
capability_matrix(self) -> 'Mapping[str, bool]'
```

Return the explicit allowed/denied capability map.

### DriverFoundry.validate\_driver

```
validate_driver(self, source: 'str') -> 'DriverFoundryResult'
```

Public operational method on this integration surface.

### DriverFoundry.compile\_driver

```
compile_driver(self, source: 'str') -> 'DriverFoundryResult'
```

Public operational method on this integration surface.

### DriverFoundry.audit\_driver

```
audit_driver(self, package: 'BytecodePackage') -> 'DriverFoundryResult'
```

Public operational method on this integration surface.

### DriverFoundry.test\_driver

```
test_driver(self, package: 'BytecodePackage', fixtures: 'Mapping[str, Any]') -> 'DriverFoundryResult'
```

Public operational method on this integration surface.

### DriverFoundry.propose\_driver

```
propose_driver(self, source: 'str', *, fixtures: 'Mapping[str, Any] | None' = None) -> 'DriverFoundryResult'
```

Validate, compile, audit, and optionally fixture-test a TDDL driver proposal in one non-authoritative flow.

### DriverFoundry.submit\_candidate

```
submit_candidate(self, package: 'BytecodePackage', *, registry: 'DriverRegistry', vm_result: 'DriverVMResult | None' = None)
```

Submit the request through the existing policy/authority layer; does not bypass authority boundaries.

## DriverVMRuntime

```
DriverVMRuntime(*, max_instructions: 'int' = 1024, max_cost: 'int' = 100000) -> 'None'
```

### DriverVMRuntime.load

```
load(self, package: 'BytecodePackage') -> 'VMLoadedPackage'
```

Load or prepare an object for later execution.

### DriverVMRuntime.execute

```
execute(self, inputs: 'Mapping[str, Any] | None' = None) -> 'DriverVMResult'
```

Execute the currently loaded deterministic bytecode package against a bounded input snapshot.

## DriverVMSkeleton

```
DriverVMSkeleton(*, max_instructions: 'int' = 1024, max_cost: 'int' = 100000) -> 'None'
```

Audit-only VM skeleton. execute() intentionally returns execution-disabled evidence.

### DriverVMSkeleton.load

```
load(self, package: 'BytecodePackage') -> 'VMLoadedPackage'
```

Load or prepare an object for later execution.

### DriverVMSkeleton.execute

```
execute(self, _inputs: 'Mapping[str, Any] | None' = None) -> 'DriverVMResult'
```

Execute the bounded operation.

## DriverRuntimeManager

```
DriverRuntimeManager(*, policy: 'RuntimeManagerPolicy | None' = None) -> 'None'
```

### DriverRuntimeManager.execute\_package

```
execute_package(self, package: 'BytecodePackage', fixtures: 'Mapping[str, Any] | None', *, registry: 'DriverRegistry | None' = None)
```

Run a package through the controlled Runtime Manager and return complete execution evidence.

### DriverRuntimeManager.run

```
run(self, package: 'BytecodePackage', fixtures: 'Mapping[str, Any] | None', *, registry: 'DriverRegistry | None' = None)
```

Public operational method on this integration surface.

## DriverRegistry

```
DriverRegistry(signature_policy: 'SignaturePolicy | None' = None) -> 'None'
```

### DriverRegistry.add\_candidate

```
add_candidate(self, manifest: 'DriverManifest', *, test_report_hash: 'str | None' = None) -> 'DriverRecord'
```

---

Public operational method on this integration surface.

#### **DriverRegistry.require**

```
require(self, driver_id: 'str') -> 'DriverRecord'
```

Public operational method on this integration surface.

#### **DriverRegistry.attach\_signature**

```
attach_signature(self, driver_id: 'str', signature: 'str') -> 'DriverRecord'
```

Public operational method on this integration surface.

#### **DriverRegistry.approve**

```
approve(self, driver_id: 'str') -> 'DriverRecord'
```

Public operational method on this integration surface.

#### **DriverRegistry.activate**

```
activate(self, driver_id: 'str') -> 'DriverRecord'
```

Public operational method on this integration surface.

#### **DriverRegistry.retire**

```
retire(self, driver_id: 'str') -> 'DriverRecord'
```

Public operational method on this integration surface.

#### **DriverRegistry.revoke**

```
revoke(self, driver_id: 'str') -> 'DriverRecord'
```

Public operational method on this integration surface.

#### **Authority order**

Candidate submission, signature attachment, approval, and activation are separate operations. Do not collapse them into a single AI-generated action. Foundry and Studio intentionally lack direct sign/activate methods.

## 16. Driver regression, review, evidence, and performance

### DriverRegressionHarness

```
DriverRegressionHarness(*, runtime_manager: 'DriverRuntimeManager | None' = None, max_cases: 'int' = 256) -> 'None'
```

#### DriverRegressionHarness.capability\_matrix

```
capability_matrix(self) -> 'Mapping[str, bool]'
```

Return the explicit allowed/denied capability map.

#### DriverRegressionHarness.run

```
run(self, package: 'BytecodePackage', cases: 'Sequence[DriverFixtureCase | Mapping[str, Any]]', *, registry: 'DriverRegi
```

Public operational method on this integration surface.

#### DriverRegressionHarness.run\_package

```
run_package(self, package: 'BytecodePackage', cases: 'Sequence[DriverFixtureCase | Mapping[str, Any]]', *, registry: 'Dr
```

Run deterministic fixture cases and return a regression report.

### DriverBatchReviewBoard

```
DriverBatchReviewBoard(*, policy: 'BatchReviewPolicy | None' = None) -> 'None'
```

#### DriverBatchReviewBoard.capability\_matrix

```
capability_matrix(self) -> 'Mapping[str, bool]'
```

Return the explicit allowed/denied capability map.

#### DriverBatchReviewBoard.review

```
review(self, reports: 'Sequence[DriverRegressionReport | DriverReviewItem | Mapping[str, Any]]', *, reviewer_id: 'str' =
```

Public operational method on this integration surface.

#### DriverBatchReviewBoard.review\_reports

```
review_reports(self, reports: 'Sequence[DriverRegressionReport | DriverReviewItem | Mapping[str, Any]]', *, reviewer_id:
```

Review regression reports and produce a bounded batch decision; registry mutation is explicit and optional.

### EvidenceBundleExporter

```
EvidenceBundleExporter(*, export_format: 'EvidenceExportFormat | str' = ) -> 'None'
```

#### EvidenceBundleExporter.capability\_matrix

```
capability_matrix(self) -> 'Mapping[str, bool]'
```

Return the explicit allowed/denied capability map.

#### EvidenceBundleExporter.export

```
export(self, batch_report: 'DriverBatchReviewReport', *, regression_reports: 'Sequence[DriverRegressionReport] | Mapping
```

Public operational method on this integration surface.

#### EvidenceBundleExporter.export\_batch\_review

```
export_batch_review(self, batch_report: 'DriverBatchReviewReport', *, regression_reports: 'Sequence[DriverRegressionRepo
```

Create an immutable JSON-safe review/evidence bundle.

#### EvidenceBundleExporter.verify

```
verify(self, bundle: 'DriverEvidenceBundle | Mapping[str, Any] | str') -> 'EvidenceIntegrityStatus'
```

Verify integrity and return structured status.

#### EvidenceBundleExporter.verify\_bundle

```
verify_bundle(self, bundle: 'DriverEvidenceBundle | Mapping[str, Any] | str') -> 'EvidenceIntegrityStatus'
```

Verify integrity and return structured status.

### DriverVMPerformanceHarness

```
DriverVMPerformanceHarness(*, policy: 'DriverVMPerformancePolicy | None' = None, runtime_manager_policy: 'RuntimeManager
```

Opt-in performance evidence. Normal DriverVMRuntime execution does not benchmark itself.

### DriverVMPerformanceHarness.capability\_matrix

```
capability_matrix(self) -> 'Mapping[str, bool]'
```

Return the explicit allowed/denied capability map.

### DriverVMPerformanceHarness.run\_package

```
run_package(self, package: 'BytecodePackage', fixtures: 'Mapping[str, Any] | None') -> 'DriverVMPerformanceReport'
```

Public operational method on this integration surface.

### DriverVMPerformanceHarness.run

```
run(self, package: 'BytecodePackage', fixtures: 'Mapping[str, Any] | None') -> 'DriverVMPerformanceReport'
```

Public operational method on this integration surface.

### runtime\_fixture\_hash

```
runtime_fixture_hash(fixtures: 'Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.drivers.regression

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### run\_studio\_quick\_test

```
run_studio_quick_test(source: 'str', *, signer: 'str' = 'studio-admin', secret: 'bytes' = b'local-driver-studio-secret')
```

**Import:** staqtapp\_tds.drivers.studio

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### studio\_instruction\_reference

```
studio_instruction_reference() -> 'Mapping[str, Mapping[str, object]]'
```

**Import:** staqtapp\_tds.drivers.studio

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### foundry\_capability\_matrix

```
foundry_capability_matrix(policy: 'DriverFoundryPolicy | None' = None) -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.foundry

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### runtime\_manager\_capability\_matrix

```
runtime_manager_capability_matrix(policy: 'RuntimeManagerPolicy | None' = None) -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.runtime\_manager

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### regression\_harness\_capability\_matrix

```
regression_harness_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.regression

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### batch\_review\_capability\_matrix

```
batch_review_capability_matrix(policy: 'BatchReviewPolicy | None' = None) -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.review

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### evidence\_export\_capability\_matrix

```
evidence_export_capability_matrix(export_format: 'EvidenceExportFormat | str' = ) -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.evidence

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### driver\_vm\_performance\_capability\_matrix

```
driver_vm_performance_capability_matrix(policy: 'DriverVMPerformancePolicy | None' = None, *, native_runner_available: 'bool')
```

**Import:** staqtapp\_tds.drivers.performance

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### driver\_vm\_performance\_enabled

```
driver_vm_performance_enabled(env: 'Mapping[str, str] | None' = None) -> 'bool'
```

---

**Import:** `staqtapp_tds.drivers.performance`

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## 17. Driver Studio and PyQt5 integration

Driver Studio has two useful programming surfaces: the headless evidence/view-model APIs and the optional PyQt5 window. Headless modules import safely without PyQt5. Studio may prepare proposals and route review requests, but it remains outside direct approval, signing, activation, Registry mutation, storage writes, and Driver VM execution.

### DriverStudioReadOnlyConsole

```
DriverStudioReadOnlyConsole(*, exporter: 'EvidenceBundleExporter | None' = None) -> 'None'
```

#### DriverStudioReadOnlyConsole.capability\_matrix

```
capability_matrix(self) -> 'Mapping[str, bool]'
```

Return the explicit allowed/denied capability map.

#### DriverStudioReadOnlyConsole.open\_bundle

```
open_bundle(self, bundle: 'DriverEvidenceBundle | Mapping[str, Any] | str', *, selected_driver_id: 'str | None' = None)
```

Public operational method on this integration surface.

#### DriverStudioReadOnlyConsole.load\_bundle

```
load_bundle(self, bundle: 'DriverEvidenceBundle | Mapping[str, Any] | str', *, selected_driver_id: 'str | None' = None)
```

Public operational method on this integration surface.

#### DriverStudioReadOnlyConsole.render

```
render(self, bundle: 'DriverEvidenceBundle | Mapping[str, Any] | str', *, selected_driver_id: 'str | None' = None) -> 'D'
```

Public operational method on this integration surface.

### DriverStudioManualProposalBuilder

```
DriverStudioManualProposalBuilder(*, foundry: 'DriverFoundry | None' = None) -> 'None'
```

#### DriverStudioManualProposalBuilder.capability\_matrix

```
capability_matrix(self) -> 'Mapping[str, bool]'
```

Return the explicit allowed/denied capability map.

#### DriverStudioManualProposalBuilder.build\_source

```
build_source(self, task: 'StudioManualDriverTask') -> 'str'
```

Public operational method on this integration surface.

#### DriverStudioManualProposalBuilder.preview\_task

```
preview_task(self, task: 'StudioManualDriverTask') -> 'StudioManualProposalPreview'
```

Build a non-mutating preview suitable for review or UI display.

#### DriverStudioManualProposalBuilder.propose\_task

```
propose_task(self, task: 'StudioManualDriverTask', *, fixtures: 'Mapping[str, Any] | None' = None) -> 'StudioManualPropo'
```

Prepare a proposal and return structured evidence; does not grant approval or activation authority.

### DriverStudioAdminReviewActions

```
DriverStudioAdminReviewActions(*, policy: 'StudioReviewSubmissionPolicy | None' = None, readonly_console: 'DriverStudioR'
```

#### DriverStudioAdminReviewActions.capability\_matrix

```
capability_matrix(self) -> 'Mapping[str, bool]'
```

Return the explicit allowed/denied capability map.

#### DriverStudioAdminReviewActions.submit

```
submit(self, console_or_bundle: 'DriverStudioConsoleSnapshot | DriverEvidenceBundle | Mapping[str, Any] | str', actions:
```

Public operational method on this integration surface.

#### DriverStudioAdminReviewActions.submit\_actions

```
submit_actions(self, console_or_bundle: 'DriverStudioConsoleSnapshot | DriverEvidenceBundle | Mapping[str, Any] | str',
```

Submit the request through the existing policy/authority layer; does not bypass authority boundaries.

## DriverStudioAdminReviewActions.route

```
route(self, console_or_bundle: 'DriverStudioConsoleSnapshot | DriverEvidenceBundle | Mapping[str, Any] | str', actions:
```

Public operational method on this integration surface.

## StudioQtBridge

```
StudioQtBridge(*, readonly_console: 'DriverStudioReadOnlyConsole | None' = None, action_layer: 'DriverStudioAdminReviewA
```

### StudioQtBridge.capability\_matrix

```
capability_matrix(self) -> 'Mapping[str, bool]'
```

Return the explicit allowed/denied capability map.

### StudioQtBridge.shell\_state

```
shell_state(self) -> 'StudioQtShellState'
```

Public operational method on this integration surface.

### StudioQtBridge.load\_bundle

```
load_bundle(self, bundle: 'DriverEvidenceBundle | Mapping[str, Any] | str', *, selected_driver_id: 'str | None' = None)
```

Hydrate the headless/PyQt5 Studio shell from a read-only evidence bundle.

### StudioQtBridge.select\_driver

```
select_driver(self, driver_id: 'str | None') -> 'StudioQtShellState'
```

Public operational method on this integration surface.

### StudioQtBridge.hydrated\_shell\_state

```
hydrated_shell_state(self)
```

Public operational method on this integration surface.

### StudioQtBridge.hydrate\_panel

```
hydrate_panel(self, kind: 'StudioPanelKind | str')
```

Public operational method on this integration surface.

### StudioQtBridge.build\_action\_request

```
build_action_request(self, driver_id: 'str', action: 'ReviewAction | str', *, reviewer_id: 'str' = 'studio-admin', ratio
```

Public operational method on this integration surface.

### StudioQtBridge.submit\_review\_action

```
submit_review_action(self, request: 'StudioReviewActionRequest | Mapping[str, Any]', *, submitted_at: 'str' = 'undated')
```

Submit the request through the existing policy/authority layer; does not bypass authority boundaries.

### StudioQtBridge.preview\_manual\_driver\_task

```
preview_manual_driver_task(self, task: 'StudioManualDriverTask') -> 'StudioManualProposalPreview'
```

Build a non-mutating preview suitable for review or UI display.

### StudioQtBridge.propose\_manual\_driver\_task

```
propose_manual_driver_task(self, task: 'StudioManualDriverTask', *, fixtures: 'Mapping[str, Any] | None' = None) -> 'Stu
```

Prepare a proposal and return structured evidence; does not grant approval or activation authority.

### StudioQtBridge.live\_event\_bridge

```
live_event_bridge(self, *, max_events: 'int' = 256)
```

Public operational method on this integration surface.

### StudioQtBridge.live\_panel\_runtime

```
live_panel_runtime(self, *, max_events: 'int' = 256)
```

Public operational method on this integration surface.

### StudioQtBridge.review\_workflow\_console

```
review_workflow_console(self, *, max_events: 'int' = 256)
```

Public operational method on this integration surface.

### StudioQtBridge.evidence\_timeline

```
evidence_timeline(self, *, max_events: 'int' = 256)
```

Public operational method on this integration surface.

### StudioQtBridge.risk\_intelligence\_cards

```
risk_intelligence_cards(self, *, max_events: 'int' = 256)
```

Public operational method on this integration surface.

### StudioQtBridge.manual\_builder\_ui\_runtime

```
manual_builder_ui_runtime(self, *, max_events: 'int' = 256)
```

Public operational method on this integration surface.

### StudioQtBridge.export\_audit\_console

```
export_audit_console(self, *, max_events: 'int' = 256)
```

Prepare or export immutable evidence in the selected format.

### StudioQtBridge.export\_integrity\_workflow

```
export_integrity_workflow(self, *, max_events: 'int' = 256)
```

Prepare or export immutable evidence in the selected format.

### StudioQtBridge.visual\_quality\_review

```
visual_quality_review(self)
```

Public operational method on this integration surface.

## StudioLivePanelRuntime

```
StudioLivePanelRuntime(*, event_bridge: 'StudioCockpitEventBridge | None' = None, max_events: 'int' = 256, contracts: 'S
```

### StudioLivePanelRuntime.capability\_matrix

```
capability_matrix(self) -> 'Mapping[str, bool]'
```

Return the explicit allowed/denied capability map.

### StudioLivePanelRuntime.current\_state

```
current_state(self, *, include_packets: 'bool' = False) -> 'StudioPanelRuntimeState'
```

Public operational method on this integration surface.

### StudioLivePanelRuntime.consume

```
consume(self, *, include_packets: 'bool' = True) -> 'StudioPanelRuntimeState'
```

Public operational method on this integration surface.

### StudioLivePanelRuntime.load\_bundle

```
load_bundle(self, bundle: 'DriverEvidenceBundle | Mapping[str, Any] | str', *, selected_driver_id: 'str | None' = None,
```

Public operational method on this integration surface.

### StudioLivePanelRuntime.select\_driver

```
select_driver(self, driver_id: 'str | None', *, timestamp: 'str' = 'undated') -> 'StudioPanelRuntimeState'
```

Public operational method on this integration surface.

### StudioLivePanelRuntime.refresh

```
refresh(self, *, reason: 'str' = 'manual refresh', timestamp: 'str' = 'undated') -> 'StudioPanelRuntimeState'
```

Refresh the snapshot/view model from copied evidence without becoming storage authority.

### StudioLivePanelRuntime.refresh\_panel

```
refresh_panel(self, kind: 'StudioPanelKind | str', *, timestamp: 'str' = 'undated') -> 'StudioPanelRuntimeState'
```

Refresh the snapshot/view model from copied evidence without becoming storage authority.

### StudioLivePanelRuntime.submit\_review\_action

```
submit_review_action(self, request: 'StudioReviewActionRequest | Mapping[str, Any]', *, submitted_at: 'str' = 'undated')
```

Submit the request through the existing policy/authority layer; does not bypass authority boundaries.

### StudioLivePanelRuntime.preview\_manual\_driver\_task

```
preview_manual_driver_task(self, task: 'StudioManualDriverTask', *, timestamp: 'str' = 'undated') -> 'tuple[StudioManual
```

Build a non-mutating preview suitable for review or UI display.

## StudioLivePanelRuntime.propose\_manual\_driver\_task

```
propose_manual_driver_task(self, task: 'StudioManualDriverTask', *, fixtures: 'Mapping[str, Any] | None' = None, timesta
```

Prepare a proposal and return structured evidence; does not grant approval or activation authority.

## StudioLivePanelRuntime.signal\_payload

```
signal_payload(self) -> 'Mapping[str, Any]'
```

Public operational method on this integration surface.

## StudioOperationalStressHarness

```
StudioOperationalStressHarness(*, runtime: 'StudioLivePanelRuntime | None' = None, admin_control: 'AdminControl | None'
```

## StudioOperationalStressHarness.capability\_matrix

```
capability_matrix(self) -> 'Mapping[str, bool]'
```

Return the explicit allowed/denied capability map.

## StudioOperationalStressHarness.run

```
run(self, *, iterations: 'int' = 32, browser_polls: 'int | None' = None, studio_refreshes: 'int | None' = None, include_
```

Public operational method on this integration surface.

## StudioOperationalStressHarness.run\_scenario

```
run_scenario(self, scenario: 'StudioOperationalStressScenario | str', *, iterations: 'int' = 32) -> 'StudioOperationalSt
```

Public operational method on this integration surface.

## StudioOperationalStressHarness.run\_scenario\_matrix

```
run_scenario_matrix(self, scenarios: 'tuple[StudioOperationalStressScenario | str, ...] | list[StudioOperationalStressSc
```

Execute the named Browser/Studio/persistence stress scenarios and return structured evidence.

## pyqt5\_available

```
pyqt5_available() -> 'bool'
```

**Import:** staqtapp\_tds.studio\_pyqt5.availability

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## pyqt5\_availability

```
pyqt5_availability() -> 'PyQt5Availability'
```

**Import:** staqtapp\_tds.studio\_pyqt5.availability

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## require\_pyqt5

```
require_pyqt5() -> 'None'
```

**Import:** staqtapp\_tds.studio\_pyqt5.availability

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## create\_driver\_studio\_window

```
create_driver_studio_window(*, bridge: 'StudioQtBridge | None' = None, theme: 'StudioQtTheme | None' = None) -> 'DriverS
```

**Import:** staqtapp\_tds.studio\_pyqt5.main\_window

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## studio\_pyqt5\_shell\_capability\_matrix

```
studio_pyqt5_shell_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.bridge

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

## studio\_cockpit\_hydration\_capability\_matrix

```
studio_cockpit_hydration_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.hydration

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_live\_event\_bridge\_capability\_matrix

```
studio_live_event_bridge_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.live

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_live\_panel\_runtime\_capability\_matrix

```
studio_live_panel_runtime_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.runtime

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_review\_workflow\_capability\_matrix

```
studio_review_workflow_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.review\_workflow

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_evidence\_timeline\_capability\_matrix

```
studio_evidence_timeline_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.evidence\_timeline

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_risk\_intelligence\_capability\_matrix

```
studio_risk_intelligence_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.risk\_intelligence

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_manual\_builder\_ui\_runtime\_capability\_matrix

```
studio_manual_builder_ui_runtime_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.manual\_builder\_runtime

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_export\_audit\_capability\_matrix

```
studio_export_audit_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.export\_audit

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_export\_integrity\_workflow\_capability\_matrix

```
studio_export_integrity_workflow_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.export\_integrity\_workflow

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_operational\_stress\_capability\_matrix

```
studio_operational_stress_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.operational\_stress

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

```
from staqtapp_tds.studio_pyqt5 import (
    StudioQtBridge,
    create_driver_studio_window,
    pyqt5_available,
)

bridge = StudioQtBridge()
shell_state = bridge.load_bundle(evidence_bundle)

if pyqt5_available():
    window = create_driver_studio_window(bridge=bridge)
    window.show()
```

# 18. Driver and Studio operational function index

## Driver namespace

### drivers.audit

#### audit\_vm\_contract

```
audit_vm_contract(package: 'BytecodePackage') -> 'None'
```

**Import:** staqtapp\_tds.drivers.audit

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### vm\_contract\_table

```
vm_contract_table() -> 'Mapping[str, Mapping[str, Any]]'
```

**Import:** staqtapp\_tds.drivers.audit

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### drivers.bytecode

#### compile\_program

```
compile_program(program: 'TDDLProgram', *, source_material: 'str | Mapping[str, Any] | None' = None) -> 'BytecodePackage'
```

**Import:** staqtapp\_tds.drivers.bytecode

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### compile\_tddl

```
compile_tddl(source_or_program: 'str | TDDLProgram') -> 'BytecodePackage'
```

**Import:** staqtapp\_tds.drivers.bytecode

**Behavior:** Validate TDDL source and compile it into a deterministic bytecode package.

#### decompile\_to\_ir

```
decompile_to_ir(package: 'BytecodePackage') -> 'TDDLProgram'
```

**Import:** staqtapp\_tds.drivers.bytecode

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### opcode\_table

```
opcode_table() -> 'Mapping[str, Mapping[str, Any]]'
```

**Import:** staqtapp\_tds.drivers.bytecode

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### validate\_bytecode\_package

```
validate_bytecode_package(package: 'BytecodePackage') -> 'None'
```

**Import:** staqtapp\_tds.drivers.bytecode

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

### drivers.evidence

#### evidence\_export\_capability\_matrix

```
evidence_export_capability_matrix(export_format: 'EvidenceExportFormat | str' = ) -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.evidence

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### drivers.foundry

#### foundry\_capability\_matrix

```
foundry_capability_matrix(policy: 'DriverFoundryPolicy | None' = None) -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.foundry

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

## drivers.manifest

### validate\_manifest

```
validate_manifest(manifest: 'DriverManifest') -> 'None'
```

**Import:** staqtapp\_tds.drivers.manifest

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## drivers.performance

### driver\_vm\_performance\_capability\_matrix

```
driver_vm_performance_capability_matrix(policy: 'DriverVMPerformancePolicy | None' = None, *, native_runner_available: 'bool')
```

**Import:** staqtapp\_tds.drivers.performance

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### driver\_vm\_performance\_enabled

```
driver_vm_performance_enabled(env: 'Mapping[str, str] | None' = None) -> 'bool'
```

**Import:** staqtapp\_tds.drivers.performance

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## drivers.regression

### regression\_harness\_capability\_matrix

```
regression_harness_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.regression

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### runtime\_fixture\_hash

```
runtime_fixture_hash(fixtures: 'Mapping[str, Any]') -> 'str'
```

**Import:** staqtapp\_tds.drivers.regression

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## drivers.review

### batch\_review\_capability\_matrix

```
batch_review_capability_matrix(policy: 'BatchReviewPolicy | None' = None) -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.review

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

## drivers.runtime\_manager

### runtime\_manager\_capability\_matrix

```
runtime_manager_capability_matrix(policy: 'RuntimeManagerPolicy | None' = None) -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.runtime\_manager

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

## drivers.signature

### sign\_payload

```
sign_payload(payload: 'bytes', *, signer: 'str', secret: 'bytes') -> 'str'
```

**Import:** staqtapp\_tds.drivers.signature

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### verify\_signature

```
verify_signature(payload: 'bytes', signature: 'str', *, signer: 'str', secret: 'bytes') -> 'bool'
```

**Import:** staqtapp\_tds.drivers.signature

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## drivers.studio

### run\_studio\_quick\_test

```
run_studio_quick_test(source: 'str', *, signer: 'str' = 'studio-admin', secret: 'bytes' = b'local-driver-studio-secret')
```

**Import:** staqtapp\_tds.drivers.studio

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### studio\_instruction\_reference

```
studio_instruction_reference() -> 'Mapping[str, Mapping[str, object]]'
```

**Import:** staqtapp\_tds.drivers.studio

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_readonly\_capability\_matrix

```
studio_readonly_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.studio

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

## drivers.studio\_actions

### studio\_admin\_review\_capability\_matrix

```
studio_admin_review_capability_matrix(policy: 'StudioReviewSubmissionPolicy | None' = None) -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.studio\_actions

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

## drivers.studio\_builder

### studio\_manual\_builder\_capability\_matrix

```
studio_manual_builder_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.studio\_builder

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

## drivers.tddl

### instruction\_specs

```
instruction_specs() -> 'Mapping[str, InstructionSpec]'
```

**Import:** staqtapp\_tds.drivers.tddl

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### parse\_tddl

```
parse_tddl(source: 'str') -> 'TDDLProgram'
```

**Import:** staqtapp\_tds.drivers.tddl

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### validate\_tddl

```
validate_tddl(program: 'TDDLProgram') -> 'None'
```

**Import:** staqtapp\_tds.drivers.tddl

**Behavior:** Validate shape, fingerprints, limits, prerequisites, and current evidence; failures are reported or raised fail-closed according to the API contract.

## drivers.trace

### rank\_traces

```
rank_traces(traces: 'list[TraceEvidence]') -> 'list[TraceEvidence]'
```

**Import:** staqtapp\_tds.drivers.trace

**Behavior:** Rank trace IDs deterministically from scores and optional confidence/depth/age inputs.

## Driver Studio PyQt5 namespace

### drivers.studio\_builder

#### studio\_manual\_builder\_capability\_matrix

```
studio_manual_builder_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.drivers.studio\_builder

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_pyqt5.availability

#### pyqt5\_availability

```
pyqt5_availability() -> 'PyQt5Availability'
```

**Import:** staqtapp\_tds.studio\_pyqt5.availability

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### pyqt5\_available

```
pyqt5_available() -> 'bool'
```

**Import:** staqtapp\_tds.studio\_pyqt5.availability

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

#### require\_pyqt5

```
require_pyqt5() -> 'None'
```

**Import:** staqtapp\_tds.studio\_pyqt5.availability

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### studio\_pyqt5.bridge

#### studio\_pyqt5\_shell\_capability\_matrix

```
studio_pyqt5_shell_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.bridge

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_pyqt5.evidence\_timeline

#### studio\_evidence\_timeline\_capability\_matrix

```
studio_evidence_timeline_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.evidence\_timeline

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_pyqt5.export\_audit

#### studio\_export\_audit\_capability\_matrix

```
studio_export_audit_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.export\_audit

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_pyqt5.export\_integrity\_workflow

#### studio\_export\_integrity\_workflow\_capability\_matrix

```
studio_export_integrity_workflow_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.export\_integrity\_workflow

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_pyqt5.hydratation

#### manual\_builder\_form\_schema

```
manual_builder_form_schema() -> 'tuple[StudioFormField, ...]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.hydratation

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### studio\_cockpit\_hydration\_capability\_matrix

```
studio_cockpit_hydration_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.hydratation

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_pyqt5.icons

#### studio\_svg\_icon

```
studio_svg_icon(name: 'str') -> 'str'
```

**Import:** staqtapp\_tds.studio\_pyqt5.icons

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_pyqt5.live

#### studio\_live\_event\_bridge\_capability\_matrix

```
studio_live_event_bridge_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.live

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

#### studio\_panel\_refresh\_contracts

```
studio_panel_refresh_contracts() -> 'tuple[StudioPanelRefreshContract, ...]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.live

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_pyqt5.main\_window

#### create\_driver\_studio\_window

```
create_driver_studio_window(*, bridge: 'StudioQtBridge | None' = None, theme: 'StudioQtTheme | None' = None) -> 'DriverS'
```

**Import:** staqtapp\_tds.studio\_pyqt5.main\_window

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### studio\_pyqt5.manual\_builder\_runtime

#### studio\_manual\_builder\_ui\_runtime\_capability\_matrix

```
studio_manual_builder_ui_runtime_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.manual\_builder\_runtime

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

#### studio\_qt\_visual\_quality\_capability\_matrix

```
studio_qt_visual_quality_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.manual\_builder\_runtime

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

#### studio\_qt\_visual\_quality\_review

```
studio_qt_visual_quality_review(*, form_fields: 'Sequence[StudioFormField] | None' = None, source: 'str' = '', theme: 'S'
```

**Import:** staqtapp\_tds.studio\_pyqt5.manual\_builder\_runtime

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_pyqt5.operational\_stress

#### studio\_operational\_stress\_capability\_matrix

```
studio_operational_stress_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.operational\_stress

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

## studio\_pyqt5.panels.descriptors

### panel\_descriptor

```
panel_descriptor(kind: 'StudioPanelKind | str') -> 'StudioPanelDescriptor'
```

**Import:** staqtapp\_tds.studio\_pyqt5.panels.descriptors

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## studio\_pyqt5.review\_workflow

### studio\_review\_rationale\_templates

```
studio_review_rationale_templates() -> 'tuple[StudioReviewRationaleTemplate, ...]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.review\_workflow

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

### studio\_review\_workflow\_capability\_matrix

```
studio_review_workflow_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.review\_workflow

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

## studio\_pyqt5.risk\_intelligence

### studio\_risk\_intelligence\_capability\_matrix

```
studio_risk_intelligence_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.risk\_intelligence

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

## studio\_pyqt5.runtime

### studio\_live\_panel\_runtime\_capability\_matrix

```
studio_live_panel_runtime_capability_matrix() -> 'Mapping[str, bool]'
```

**Import:** staqtapp\_tds.studio\_pyqt5.runtime

**Behavior:** Return a JSON-safe capability or presentation contract for Studio integration.

## 19. Centralized Browser and local admin control

The Browser is the centralized operational view for the TDS engine and CSV/Interpole telemetry. The server is intentionally small: HTML/CSS/JS assets are packaged, `/status.json` serves snapshots, and configuration mutations require local authority, same-origin checks, and CSRF tokens.

### AdminControl

```
AdminControl(registry: 'ConfigRegistry | None' = None, auth: 'LocalAuthProvider | None' = None, audit: 'AuditLog | None'
```

#### AdminControl.status

```
status(self) -> 'dict[str, Any]'
```

Public operational method on this integration surface.

#### AdminControl.stage\_config

```
stage_config(self, candidate: 'RuntimeConfig', grant: 'ConfigGrant') -> 'RuntimeConfig'
```

Public operational method on this integration surface.

#### AdminControl.promote\_config

```
promote_config(self, grant: 'ConfigGrant') -> 'RuntimeConfig'
```

Public operational method on this integration surface.

#### AdminControl.rollback\_config

```
rollback_config(self, grant: 'ConfigGrant') -> 'RuntimeConfig'
```

Public operational method on this integration surface.

### AdminPanelServer

```
AdminPanelServer(control: 'AdminControl | None' = None, host: 'str' = '127.0.0.1', port: 'int' = 8765)
```

Local ThreadingHTTPServer wrapper. Construct with an AdminControl whose observation\_source exposes snapshot methods.

#### AdminPanelServer.serve\_forever

```
serve_forever(self)
```

Public operational method on this integration surface.

```
from staqtapp_tds.admin import AdminControl, AdminPanelServer

control = AdminControl(observation_source=fs)
server = AdminPanelServer(
    control=control,
    host="127.0.0.1",
    port=8765,
)
server.serve_forever()
```

#### Production binding

The default local authentication provider is suitable for localhost development. Do not expose the Browser beyond localhost without replacing development secrets and enforcing the surrounding network/security policy.

## 20. Configuration, native management, security, and low-level calls

### RuntimeConfig

```
RuntimeConfig(config_id: 'str', generation: 'int' = 1, created_at: 'float' = , chunk_bytes: 'int' = 65536, compression:
```

Immutable runtime configuration. Stage and promote through ConfigRegistry/AdminControl rather than modifying live settings in place.

### RuntimeConfig.next\_generation

```
next_generation(self, **changes: 'Any') -> "RuntimeConfig"
```

Public operational method on this integration surface.

### ConfigRegistry

```
ConfigRegistry(initial: 'RuntimeConfig | None' = None)
```

### ConfigRegistry.active

```
active(self) -> 'RuntimeConfig'
```

Public operational method on this integration surface.

### ConfigRegistry.stage

```
stage(self, candidate: 'RuntimeConfig') -> 'RuntimeConfig'
```

Public operational method on this integration surface.

### ConfigRegistry.promote

```
promote(self, candidate: 'RuntimeConfig | None' = None) -> 'RuntimeConfig'
```

Public operational method on this integration surface.

### ConfigRegistry.rollback

```
rollback(self) -> 'RuntimeConfig'
```

Public operational method on this integration surface.

### ConfigRegistry.snapshot

```
snapshot(self) -> 'dict[str, Any]'
```

Return a JSON-safe snapshot.

### NativeEngineManager

```
NativeEngineManager(*, expected_abi: 'int' = 1) -> 'None'
```

### SecureParams

```
SecureParams(values: 'Mapping[str, str]' = ) -> None
```

Security-related parameter value object.

## 20.1 Advanced root-level functions

### encode\_header

```
encode_header(ts_create: 'int', ts_mod: 'int', flags: 'int', fmt_id: 'int', subdir_count: 'int', entry_count: 'int') ->
```

**Import:** staqtapp\_tds.tds\_filesystem

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### decode\_header

```
decode_header(raw: 'bytes') -> 'dict'
```

**Import:** staqtapp\_tds.tds\_filesystem

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### load\_manifest

```
load_manifest(folder: 'Path', *, inherit: 'bool' = True) -> 'ManifestPolicy'
```

**Import:** staqtapp\_tds.manifest

**Behavior:** Load an existing committed artifact or report and reconstruct its typed representation.

### write\_default\_manifest

```
write_default_manifest(folder: 'Path', *, overwrite: 'bool' = False) -> 'Path'
```

**Import:** staqtapp\_tds.manifest

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### route\_id\_for

```
route_id_for(stamp: 'str', path: 'str' = '') -> 'int'
```

**Import:** staqtapp\_tds.srz

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### classify\_latency

```
classify_latency(actual_ns: 'int', expected_ns: 'int', soft_limit_ns: 'int' = 0, hard_limit_ns: 'int' = 0) -> 'LatencyBu'
```

**Import:** staqtapp\_tds.latency

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### latency\_ratio

```
latency_ratio(actual_ns: 'int', expected_ns: 'int') -> 'float'
```

**Import:** staqtapp\_tds.latency

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### known\_result\_codes

```
known_result_codes() -> 'tuple[str, ...]'
```

**Import:** staqtapp\_tds.result

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### is\_known\_result\_code

```
is_known_result_code(code: 'TDSResultCode | str') -> 'bool'
```

**Import:** staqtapp\_tds.result

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### result\_info

```
result_info(code: 'TDSResultCode | str') -> 'TDSResultInfo | None'
```

**Import:** staqtapp\_tds.result

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### choose\_variable\_kind

```
choose_variable_kind(value: 'Any') -> 'PayloadKind'
```

**Import:** staqtapp\_tds.serializers

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### content\_hash\_bytes

```
content_hash_bytes(raw: 'bytes', algorithm: 'str' = 'sha256') -> 'str'
```

**Import:** staqtapp\_tds.serializers

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

### dumps\_pickle

```
dumps_pickle(value: 'Any', policy: 'PicklePolicy | None' = None) -> 'bytes'
```

**Import:** staqtapp\_tds.tds\_pickle

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## loads\_pickle

```
loads_pickle(raw: 'bytes', policy: 'PicklePolicy | None' = None) -> 'Any'
```

**Import:** staqtapp\_tds.tds\_pickle

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## pickle\_policy\_snapshot

```
pickle_policy_snapshot(policy: 'PicklePolicy | None' = None) -> 'dict[str, Any]'
```

**Import:** staqtapp\_tds.tds\_pickle

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## get\_default\_serialization\_manager

```
get_default_serialization_manager() -> 'SerializationManager'
```

**Import:** staqtapp\_tds.serialization.manager

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## query\_requires\_selector

```
query_requires_selector(**selectors) -> 'TDSResult'
```

**Import:** staqtapp\_tds.cluster

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## create\_spiral\_run

```
create_spiral_run(root, run_id: 'str', *, problem: 'Any | None' = None, problem_id: 'str' = '', description: 'str' = '',
```

**Import:** staqtapp\_tds.spiral.run

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## verify

```
verify(target: 'Any') -> 'HealthReport'
```

**Import:** staqtapp\_tds.verify

**Behavior:** Run the TDS health verifier against a supported target and return a structured report.

## estimate\_pressure

```
estimate_pressure(counters: 'Mapping[str, int]', *, queue_depth: 'int' = 0, snapshot_lag: 'int' = 0, swiss_probe_pressur
```

**Import:** staqtapp\_tds.asi

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## native\_diagnostics\_available

```
native_diagnostics_available() -> 'bool'
```

**Import:** staqtapp\_tds.diagnostics

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## native\_diag\_snapshot

```
native_diag_snapshot(*, event_limit: 'int' = 32) -> 'NativeDiagnosticSnapshot'
```

**Import:** staqtapp\_tds.diagnostics

**Behavior:** Control or inspect the bounded native diagnostic side channel; not a storage data operation.

## native\_diag\_reset

```
native_diag_reset() -> 'bool'
```

**Import:** staqtapp\_tds.diagnostics

**Behavior:** Control or inspect the bounded native diagnostic side channel; not a storage data operation.

## native\_diag\_set\_enabled

```
native_diag_set_enabled(enabled: 'bool') -> 'bool'
```

**Import:** staqtapp\_tds.diagnostics

**Behavior:** Control or inspect the bounded native diagnostic side channel; not a storage data operation.

## native\_diag\_mark\_degraded

```
native_diag_mark_degraded(degraded: 'bool' = True) -> 'bool'
```

**Import:** staqtapp\_tds.diagnostics

**Behavior:** Control or inspect the bounded native diagnostic side channel; not a storage data operation.

## native\_diag\_emit

```
native_diag_emit(event: 'DiagnosticEvent | int', value_a: 'int' = 0, value_b: 'int' = 0) -> 'bool'
```

**Import:** staqtapp\_tds.diagnostics

**Behavior:** Control or inspect the bounded native diagnostic side channel; not a storage data operation.

## calculate\_pressure\_snapshot

```
calculate_pressure_snapshot(counters: 'Mapping[str, Any]', *, native_counters: 'Mapping[str, Any] | None' = None, perform...
```

**Import:** staqtapp\_tds.pressure

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## build\_recovery\_plan

```
build_recovery_plan(pressure: 'Mapping[str, Any] | None', *, native_diagnostics: 'Mapping[str, Any] | None' = None, perform...
```

**Import:** staqtapp\_tds.recovery

**Behavior:** Convert pressure/diagnostic evidence into a non-mutating recovery plan.

## get\_native\_manager

```
get_native_manager() -> 'NativeEngineManager'
```

**Import:** staqtapp\_tds.native.manager

**Behavior:** Public operational helper. Use the exact signature and returned typed object shown here; inspect the returned status/ok fields before continuing.

## native\_status\_result

```
native_status_result() -> 'TDSResult'
```

**Import:** staqtapp\_tds.native.manager

**Behavior:** Return a non-halting result describing native engine availability and state.

## native\_capabilities\_result

```
native_capabilities_result() -> 'TDSResult'
```

**Import:** staqtapp\_tds.native.manager

**Behavior:** Return a non-halting result containing the native capability matrix.

## 21. Integration recipes

### 21.1 Persist an AI checkpoint atomically

```
checkpoint = fs.makedirs("/checkpoints/current")
checkpoint.write_json("planner", planner_state, overwrite=True)
checkpoint.write_json("memory", memory_state, overwrite=True)
checkpoint.write_text("summary", summary, overwrite=True)

health = verify(fs)
if health.status != "healthy":
    raise RuntimeError(health)

TDSPersistence("./state").flush(fs, parallel_nodes=True)
```

### 21.2 Read many model inputs from one .tds node

```
with_reader = TDSReader("./state/agent__inputs.tds")
try:
    values = with_reader.read_many([
        "/agent/inputs/system_prompt",
        "/agent/inputs/tool_policy",
        "/agent/inputs/context_window",
    ])
finally:
    with_reader.close()
```

### 21.3 CSV evidence chain helper

```
from staqtapp_tds.csv_layer import (
    commit_csv_interpole_determinant_vector_report,
    commit_csv_interpole_timeline_report,
    commit_csv_interpole_timeline_ring_report,
    commit_csv_kernel_performance_gate_report,
    commit_csv_kernel_readiness_contract_report,
    commit_csv_native_row_anchor_kernel_report,
    commit_csv_native_scan_kernel_prototype_report,
    commit_csv_native_storage_artifacts,
    commit_csv_native_storage_revalidation_report,
    commit_csv_storage_adapter_replay_report,
    commit_csv_storage_bridge_manifest,
)

def prepare_semantic_ready_csv(directory, csv_id, *, chunk_size=7):
    reports = (
        commit_csv_storage_bridge_manifest(directory, csv_id),
        commit_csv_storage_adapter_replay_report(directory, csv_id),
        commit_csv_native_storage_artifacts(directory, csv_id),
        commit_csv_native_storage_revalidation_report(directory, csv_id),
        commit_csv_interpole_timeline_report(directory, csv_id),
        commit_csv_interpole_determinant_vector_report(directory, csv_id),
        commit_csv_interpole_timeline_ring_report(directory, csv_id),
        commit_csv_kernel_readiness_contract_report(
            directory, csv_id, chunk_size=chunk_size
        ),
        commit_csv_native_scan_kernel_prototype_report(
            directory, csv_id, chunk_size=chunk_size
        ),
        commit_csv_native_row_anchor_kernel_report(
            directory, csv_id, chunk_size=chunk_size
        ),
        commit_csv_kernel_performance_gate_report(directory, csv_id),
    )
    failures = [r for r in reports if not r.ok]
    if failures:
        raise RuntimeError(f"CSV evidence chain failed: {failures}")
    return reports
```

## 22. API selection matrix

| Programming need                  | Use                                               | Avoid                                                |
|-----------------------------------|---------------------------------------------------|------------------------------------------------------|
| Store arbitrary application state | TDSDirectory.write_result / read_result           | Direct mutation of internal _entries.                |
| Store interoperable external data | write_json / write_text / raw bytes               | Untrusted pickle payloads.                           |
| Persist a tree                    | TDSPersistence.flush                              | Writing .tds headers/slots manually.                 |
| Fast persisted lookup             | TDSReader.read / read_many                        | Loading and decoding the entire file yourself.       |
| Append model context              | stalkvar("~name", value)                          | Inventing uncontrolled sibling names.                |
| Rank trace candidates             | rank_traces / rank_trace_run                      | Unstable ad-hoc sorting when tie behavior matters.   |
| Import CSV                        | import_csv_bytes / import_csv_file                | One TDS entry per cell.                              |
| Publish CSV artifact family       | CSV transaction or commit_* calls                 | Partial manual writes under final keys.              |
| Inspect native status             | native_status_result / native_capabilities_result | Assuming the compiled extension exists.              |
| Create driver proposal            | DriverFoundry.propose_driver                      | Executing generated Python source.                   |
| Execute trusted bytecode          | DriverRuntimeManager.execute_package              | Giving Studio or Browser direct execution authority. |
| Show operational state            | AdminPanelServer / StudioQtBridge                 | Walking storage locks from a UI refresh loop.        |

---

## 23. Stable boundaries to preserve

- Storage hot path performs narrow mechanical work; observers consume copied counters, events, and immutable snapshots.
- TDSResult-first APIs are the default for long-running AI services where expected failures must not halt the process.
- CSV original bytes remain preserved; canonical exports and all later evidence are derived artifacts.
- Native CSV paths are parity-gated against the Python reference path.
- Semantic IR requires explicit caller declarations and authorization. TDS validates process and lineage, not truth.
- Driver Foundry and Studio cannot sign or activate drivers. Runtime Manager, Review Board, Registry, and signature policy retain authority.
- Browser and Studio do not bypass storage, policy, or cryptographic boundaries.

### Current Semantic IR scope

v3.5.2 is an in-memory review contract for candidates, lifecycle transitions, and atomic batches. Durable Semantic IR ledger persistence is not part of the current release.

## 24. Validation basis and compatibility

This guide was built against the Staqtapp-TDS v3.5.2 source delivery. The release validation record reports 683 passed / 11 skipped in fallback mode, 694 passed with the native extensions, 61 packaged Semantic IR tests passed, fresh-archive release checks passed, and packaged native builds passed.

The code snippets in the README and the primary storage/variable/ranking/CSV snippets in this guide were executed against the same source tree during documentation validation.

| Surface                                   | Compatibility expectation                                                                                                 |
|-------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| Python fallback                           | Core storage, persistence, CSV reference scans, semantics, Driver systems, Browser, and headless Studio remain available. |
| Native index extension                    | Optional acceleration and native diagnostics/capability evidence.                                                         |
| CSV native scan extension                 | Optional parity-gated scan and row-anchor acceleration.                                                                   |
| PyQt5                                     | Optional graphical Driver Studio; headless Studio APIs remain import-safe without it.                                     |
| Historical v3.5.0/v3.5.1 semantic objects | v3.5.2 validates compatible lineage and records the upgrade in newly produced values.                                     |

---

## 25. Final programmer checklist

- Use TDSDirectory result-first calls at application boundaries.
- Choose explicit text/JSON/raw formats for externally supplied data.
- Flush complete directory trees through TDSPersistence and close TDSReader objects.
- Treat validation/replay reports as gates, not decorative telemetry.
- Keep CSV source immutable and publish only complete artifact families.
- Do not force native mode unless native absence must be a failure.
- Keep semantic declarations, reviewer authorization, and evidence references explicit.
- Route driver execution through Runtime Manager and driver authority through Review Board/Registry/signature policy.
- Use the centralized Browser and Studio as observation/review surfaces, not direct storage authority.

---

For exhaustive passive class fields and complete class-by-class inventory, open `Staqtapp_TDS_API_Surface_Reference.pdf`.

# Addendum: Persistence Safety and Recovery Policy

**Applies from TDS v3.5.3 development series.** This addendum defines the programmer-facing data-survival contract before the immutable-generation disk format is activated.

## Critical warning

**Reducing generation retention lowers storage use but also reduces recoverable history.** Relaxed durability may improve throughput, but recently acknowledged writes can be lost after abrupt process, operating-system, or power failure. Internal TDS generations are local recovery mechanisms; they are not substitutes for off-device backups.

## Safe default

Use the production-safe preset unless the stored data is demonstrably reproducible and the application accepts a documented loss window.

```
policy = PersistencePolicy.production_safe( retained_generations=2, )
```

Atomic generation construction is mandatory. The API does not expose an `atomic_generations=False` option. A failed commit may lose the new write, but it must never destroy the previous committed generation.

## Policy fields and risks

| Field                                   | Meaning                                              | Principal risk                                          |
|-----------------------------------------|------------------------------------------------------|---------------------------------------------------------|
| <code>durability="durable"</code>       | Acknowledge after required durability barriers.      | Higher commit latency; strongest local crash guarantee. |
| <code>durability="group_durable"</code> | Share durability barriers inside an explicit window. | Writes may remain non-durable during that window.       |
| <code>durability="relaxed"</code>       | Atomic visibility; OS flushing may be deferred.      | Recently acknowledged writes may be lost.               |
| <code>retained_generations=1</code>     | Keep only current state after cleanup.               | No historical rollback remains after cleanup.           |
| <code>retained_generations&gt;=2</code> | Keep current plus prior states.                      | Greater disk consumption.                               |
| <code>cleanup="immediate"</code>        | Reclaim eligible history promptly.                   | Less diagnosis and manual-recovery time.                |
| <code>cleanup="background"</code>       | Reclaim outside foreground commit.                   | Temporary extra disk use.                               |
| <code>cleanup="manual"</code>           | Engineer explicitly prunes history.                  | Disk growth if maintenance is neglected.                |

# Runtime verification: what protection is active now?

Applications should inspect runtime status instead of assuming that deployment settings match source code.

```
status = store.persistence.status() status.durability status.retained_generations
status.atomic_generations status.external_backup_configured status.current_generation
status.last_verified_generation status.recovery_fallback_active
```

A recovery fallback must never be silent. When fallback is active, status must identify the requested generation, the generation actually mounted, and the recovery reason.

## Internal recovery is not external backup

Generation retention protects against interrupted commits and permits local rollback to retained valid states. It does not protect against device loss, host destruction, filesystem-wide corruption, theft, ransomware with filesystem access, or deletion of the entire mount. TDS therefore reports external backup configuration separately.

## Destructive maintenance

Pruning and unpinning can permanently remove rollback points. Production APIs should require explicit acknowledgement when an operation reduces retained history below the recommended default. Pinned generations must never be removed by automatic cleanup.

## Performance contract

Generation policy is resolved at mount time and must not enter the normal read hot path. The active manifest is cached. Checksums are calculated while bytes are written. Cleanup normally occurs outside the foreground commit path. Later segment sharing may reduce physical writes without weakening the logical recovery contract.

## Release maturity note

In v3.5.3-dev1 these types define and test the public contract. The v2 disk format remains unchanged until immutable generation writing, atomic promotion, startup recovery, fault injection, retention, and migration are implemented and qualified together.

# Addendum: Generation Retention and Cleanup Safety

**Applies from TDS v3.5.3 development series.** Cleanup is a maintenance subsystem. Recovery never deletes rejected evidence.

## Persistent protection

A verified generation can be pinned across process restarts. Pin markers are independent of generation metadata, so cleanup does not trust an unverified manifest to decide protection.

```
store.pin(generation_id)
store.list_pins()
store.unpin(generation_id, acknowledge_reduced_recovery=True)
```

## Destructive acknowledgement

Manual pruning refuses to delete any eligible generation unless `acknowledge_reduced_recovery=True` is supplied. Unpinning requires the same deliberate acknowledgement because it can expose that generation to later deletion.

```
report = store.prune(keep=2, acknowledge_reduced_recovery=True)
```

## Restartable cleanup

TDS durably writes a bounded cleanup plan before deletion. For every candidate it recomputes current and pin protection, atomically moves the generation into a quarantine directory, then removes it. An interrupted plan can be resumed safely.

```
report = store.resume_cleanup()
```

## Safety contract

| Invariant              | Behavior                                  |
|------------------------|-------------------------------------------|
| Current generation     | Always protected                          |
| Pinned generation      | Always protected                          |
| New pin during cleanup | Observed before that candidate is deleted |
| Interrupted deletion   | Resumed idempotently from CLEANUP.json    |
| Invalid cleanup plan   | Fails closed; no generation is deleted    |
| Recovery evidence      | Never deleted by recovery                 |

**Risk:** pruning and unpinning can permanently remove rollback history. Local generations do not protect against host or disk loss; maintain an off-device backup for critical state.