SMT_AllinOne + Roman Essentials v0.1 — NT8 Chassis Specification Template

Shared mechanics for every trader entry plugin (Nitro, Dhesi, Lanto, ICT2023, ORB, ...)

David Oliver trading farm (DSG) — written by Grok Bot

2026-10-01 (Australia/Brisbane)

CHASSIS SPECIFICATION TEMPLATE v0.1 (NinjaTrader 8 target)

Product	**SMT_AllinOne + Roman Essentials v0.1 (chassis) + one ENTRY plugin per trader**
Production target	**NinjaTrader 8** (NinjaScript C#). The Python stack is the reference twin and the research engine
Template status	**TEMPLATE. The NT8 C# skeleton has NOT been compiled or run in NinjaTrader 8.** It was syntax-checked on the box only, against stub NT types (see §11.4). Dev PC only; nothing goes live
Written	2026-10-01, Brisbane (UTC+10)
Python reference	`smt-python/smt_strategy/essentials_host/` (`engine.py`, `plugin.py`, `types.py`, `loader.py`); `smt_strategy/allinone/` + CLI `run_smt_allinone_roman.py`
Config	`config/essentials_host/default.yaml`, `essentials_params_full.json/.yaml` (294 params), `essentials_groups.json`
Docs	`docs/SMT_ALLINONE_ROMAN_PARAMS.md` (294 params / 11 categories / 37 groups), `docs/SMT_ALLINONE_ROMAN_PARITY.md`
Contract	`entry_plugins/nitro_crt_v1/contracts/ENTRY_SIGNAL_CONTRACT.md` (schema **1.0**)
Worked plugin	`entry_plugins/nitro_crt_v2/` (`BUILD_PLAN.md`, crt_v2_m1, gaps G1–G7)
Existing NT8 code read	`smt-python/ninja/SMT_Advanced.cs` (lean SMT port), `entry_plugins/nitro_crt_v1/ninja/NitroCRT_Entry_v1.cs` (entry-only stub), `/workspace/pc3_v683_fillfix_install/SetupHunter_v6_8_3_FILL_FIX_NT8.cs` (the NT8 code behind the TDL "wired" statuses), TDL `zz-strategy-params.json` (read-only). The full **SMT Strategy V3 `AllinOneStrategy` lives on PC2** and was **not** read

Honesty rule. Every behaviour below is taken from the code or docs named in the text. Anything the code does not do is marked **GAP** or **TODO**. "Python" means Essentials v0.1 on the box. "NT8 template" means the C# skeleton shipped with this document.

Contents. 1 Purpose and boundary · 2 Data feed · 3 Bar alignment and multi-TF resampling · 4 Plugin interface · 5 Entry signal contract · 6 Order and position handling · 7 Risk engine · 8 Instrument specs · 9 Param categories and groups · 10 Backtest harness and reporting · 11 Validation, school gate and promotion · 12 Known gaps in v0.1 · 13 Fill-in trader section (+ 13-EX Nitro CRT v2 worked example) · Appendix A Package layout and NT8 import · Appendix B Change log

How to use this template for a new trader. Copy §13 and `trader_plugin_template.yaml`, then copy `Plugins/BlankTraderPlugin.cs` and `SMT_AR_BlankTrader.cs`. Fill in the trader section first. Sections 1–12 are shared and should not be edited per trader. If a trader needs something the chassis does not offer, add it to §12 as a gap for David's go-ahead; do not work around it inside the plugin.

1. Purpose and boundary: the plugin is entry-only; the chassis owns OM

1.1 Purpose

The chassis is the one shared order-management (OM) and risk layer, called **Roman Essentials / ZZ**. About 20 trader entry plugins plug into it. Each plugin turns one trader's setup rules into **ENTRY signal contracts**. The chassis decides whether to take a signal and then manages the trade: size, stop, target, break-even (BE), trail, caps, session and EOD.

1.2 Hard boundary (ENTRY_SIGNAL_CONTRACT.md §1; VERSION_LOG; BUILD_PLAN §0)

Actor	Owns	Must NOT
Entry plugin (C#: `IEntryPlugin`; Python: `EntryPlugin.emit`)	Structure detection, filters, trigger confirmation, invalidation level as `stop_hint`, target levels as `target_hint`, pack `grade`, optional `confluence_lineup`	Size contracts, trail stops, enforce max trades/day, pick account risk %, run its own session gates for risk, call `EnterLong/SetStopLoss/ExitLong`
Chassis = Roman Essentials / ZZ OM (C#: `SMT_AllinOne_Roman_Chassis`; Python: `Essentials`)	Position size, final stop (may widen or tighten versus `stop_hint`), TP ladder, session allow/deny, max trades, BE, trailing, EOD, prop consistency	Invent a per-pack OM stack; ignore the pack's `direction` / `setup_id`

- Entry logic is **never merged** into Essentials, `SMT_Advanced` or `AllinOneStrategy` (VERSION_LOG).
- Python `signal_contract.validate_signal` (nitro_crt_v2) rejects any contract that carries `quantity`, `size`, `contracts`, `trail` or `max_trades`. The C# `EntrySignal.ValidateStrict()` mirrors this for keys in `meta`.
- A setup-eligibility rule (for example Nitro F7, "only the day's first CRT is a valid setup") is plugin logic. A **trade counter** is OM logic and is mapped onto chassis params (`UseOneTradeMode`, `OneTradeMax`).
- Config merge (`loader.py`): **the entry plugin wins** on conflict for trading knobs. The plugin-owned Advanced sections are `core, entry, confluence_score, data, risk, session, htf_bias`. Host-only and unwired params are kept as passthrough.

1.3 NT8 shape

One NT8 **strategy per trader setup**, for example `SMT_AR_NitroCRT_v2 : SMT_AllinOne_Roman_Chassis`. The abstract base holds the data feed, bar alignment, OM and risk engine, plus all Essentials `NinjaScriptProperty` params. The concrete class supplies `SetPluginDefaults()` (Name and plugin-intent defaults) and `CreatePlugin()` (builds the `IEntryPlugin` from its own properties). The plugin is a plain C# class with no access to orders.

> BUILD_PLAN §8 offers an alternative shape: an indicator plus an AddOn bridge that sends the contract to an external ZZ strategy. The template uses **base class + plugin object in one strategy**, because the Python twin also runs host and plugin in one process. **TODO:** David to confirm the shape before any compile on a Dev PC.

2. Data feed

2.1 Sources (box, research)

Instrument	File	Range (per BUILD_PLAN §5)	Notes
NQ 1m (proxy for MNQ)	`/workspace/setuphunter_py/data/nq_front_month_m1.csv`	2025-07-01 → 2026-09-14 05:58 ET	From 2025-09, about 50 % of prices are 1-dp rounded (x.2/x.8). Snap to 0.25 (error ≤ 0.05 pt)
ES 1m (proxy for MES)	`es_front_month_m1.csv` (+ per-contract `es1225/0326/0626/0926_m1.csv`)	2025-09-10 → 2026-09-11	Roll-mix days in `data_prep.ES_EXCLUDE_DATES`
GC 1m (proxy for MGC)	`gc_front_month_m1.csv`	2025-12-31 → 2026-09-11 (≈ 8.3 months)	Single roll-jump bars in `GC_EXCLUDE_DATES`
Common 3-symbol window	—	2025-12-31 → 2026-09-11	Above the 3-month gate
`*_ntclock.csv` variants	same folder	—	**TODO:** provenance and clock not verified for this template; do not use without checking
Stocks 1m	PC2 `C:\GROK_BOT_TradingFarm` (NT 1m, Databento 15s/30s/tick)	—	**Must be built from trade prints only** (see 2.4)
True micro series (MNQ/MES/MGC)	—	**missing**	Full-size proxies have identical prices; only volume differs
Sub-minute futures	—	**missing**	Needed to settle entry-bar ambiguity (§10.3)

NT8 production feed. Live and Strategy Analyzer data come from the NT8 data connection and historical server. Use a **1-Minute primary series** on the traded instrument (the chassis disables itself otherwise, checked in `State.DataLoaded` via `BarsPeriod.BarsPeriodType == Minute && BarsPeriod.Value == 1`). An optional `AddDataSeries(BarsPeriodType.Tick, 1)` management series (property `UseTickManagementSeries`) follows the SetupHunter v6.8.3 pattern.

2.2 Timezone and bar labelling (critical)

Source	Stamp	Convert to
Box futures CSVs (`datetime` column)	**UTC, labelled by bar END** (verified in BUILD_PLAN §5 by RTH-open volume spikes: 14:31 UTC in January, 13:31 UTC in July). The `*_STITCH.json` claim "session-local wall clock" is **wrong** for these CSVs	**ET bar START**: `start_utc = end_utc − 1 min`, then `tz_convert("America/New_York")` (`nitro_crt_v2/python/data_prep.load_m1_utc_end`)
NT8 `Time[0]`	Bar **END**, in the NT/PC time zone (farm PCs = Brisbane)	`TimeZoneInfo.ConvertTime(Time[0], chartZone, "Eastern Standard Time")`, then `−1 min` gives the ET bar start (`SMT_AllinOne_Roman_Chassis.ToEt`). DST-safe
Python Essentials `_as_dt`	**A naive datetime is treated as UTC**	**GAP:** passing naive ET datetimes into `accept_signal`/`on_bar` silently shifts the session gate by 4–5 h. Always pass tz-aware timestamps

- All session rules are in **ET** (`sessions.tz: America/New_York`). The chassis property `SessionTimeZoneId` defaults to `"Eastern Standard Time"` (Windows id). `ChartTimeZoneId = ""` means Windows local.
- `SMT_Advanced.cs` defaults `ConvertChartTimeToNy=false`, using NT wall clock as-is for parity with the CSV stamps. The chassis template **always converts**. **TODO:** confirm on a Dev PC that NT's General → Time zone option equals Windows local, or set `ChartTimeZoneId`.
- `emitted_at` must be ISO-8601 **with offset** (contract §2). `EntrySignal.SetEmittedAt(etClock, etZone)` builds it. Python v2 raises if `emitted_at` is naive.

2.3 Roll handling

- The box files are **front-month stitches of discrete contract text exports** (`NQ_FRONT_MONTH_STITCH.json`: "NOT NT continuous… Do not invent bars"). Roll rule: session_day = (datetime − 1 min).date; during the overlap prefer the contract with higher daily volume.
- **Roll weeks must be excluded.** Dates checked 2026-10-01 (`data_prep.py`):
  - NQ: 2025-12-10..12, 2026-03-11..13, 2026-03-15, 2026-06-11..12
  - ES: 2025-12-10..12, 12-14, 12-15; 2026-03-11..13, 03-15, 03-16; 2026-06-07..12, 06-14, 06-15 (found by a jump scan: > 0.4 % inside 2 min)
  - GC: 2026-01-21, 02-02, 03-25, 05-25, 07-22 (single roll-jump bars)
- A date is excluded if **either** its ET date or its UTC-end date is on the list.
- **NT8:** use the specific contract month or NT's merge policy. **GAP/TODO:** the chassis has no roll-date exclusion list. Strategy Analyzer runs on `MNQ ##-##` with back-adjust/merge settings must be documented per run, and roll days must be excluded in post-processing until a `RollExcludeDates` param is added.

2.4 Stock 1m: trade prints only

Stock 1m bars **must be aggregated from trade prints only** (`action == 'T'`). Quote, add, cancel and modify messages must not move OHLC (this is the action=='T' bug fix). **TODO:** the fixing code was not found in `smt-python` or `setuphunter_py`; link its path here. Stock instrument specs are a **GAP** (§8).

2.5 Data-quality checks (run before every backtest; record results in the run's RESULTS.md)

#	Check	Rule / source
DQ1	Clock	Confirm UTC bar-end by the RTH-open volume spike (13:31/14:31 UTC). Never trust stitch-JSON clock notes
DQ2	Tick grid	Snap O/H/L/C to the tick size, then re-derive `high=max(OHLC)` and `low=min(OHLC)` (`load_m1_utc_end(snap_to_tick=True)`)
DQ3	Duplicates	Drop duplicate timestamps (keep first) and sort
DQ4	Roll weeks	Apply the exclude lists in 2.3, and re-scan for new rolls (open-vs-prev-close jump > 0.4 % inside 2 min)
DQ5	Gaps	Track HTF `fill_frac` and reject HTF bars with < `f1_htf_min_fill_frac` (0.5) of their minutes
DQ6	Coverage	Span ≥ ~90 days for the school gate; list missing weeks (for example the Nitro video week 2026-09-14..18 is missing)
DQ7	Holidays	**GAP:** no CME holiday / early-close calendar. Nitro M1 traded the Labor Day and Memorial Day sessions
DQ8	Hash	Record the sha256 of each data file on the box and the PC (the M1 smoke did this)
DQ9	NT8	Log `Instrument.MasterInstrument.TickSize/PointValue` at DataLoaded and assert against §8 (the chassis does this)

3. Bar alignment and multi-TF resampling

3.1 Rule: build HTF bars in code from the 1m primary, on the ET clock grid

- Supported TFs: **5, 10, 15, 20, 30 and 60 minutes** (each must divide 60; `params.validate`). A 15m bar covers `[hh:00, hh:15)`. The **1H bar is the clock hour** `[hh:00, hh+1:00)` (the CME clock hour), not a 9:30-anchored hour.
- Bars are keyed by **ET bar start**. An HTF bar closes when its last minute arrives (`closedWith`) **or** when a 1m bar from a later bucket arrives (`closedBefore`, a gap), whichever comes first. **No look-ahead:** tested in `test_bars.py` across DST and gaps, with parity against vectorised `resample_clock`.
- Eligibility: bars in `closedBefore` are usable on the current 1m bar. Bars in `closedWith` close at `ts + 1 min` and are usable **from the next bar** (`plugin.on_bar` ordering).
- `fill_frac = n / tf`. A plugin may require a minimum (Nitro: 0.5).

3.2 NT8 implementation

- `ClockAggregator` (C#, `SMT_AllinOne_Roman_Contracts.cs`) is a line-by-line port of `bars.py`. The chassis creates one per TF listed in `plugin.RequiredTimeframesMinutes` and feeds every primary 1m bar (`BarsInProgress == 0`).
- **Do not use `AddDataSeries(BarsPeriodType.Minute, 60)` for signals.** NT minute bars are stamped at bar **end** and anchored to the **session template**, so an RTH template gives 9:30-anchored hours. Only ETH CME templates (18:00 ET start) happen to land on the clock (BUILD_PLAN §8). The property `AddAdvisoryHtfSeries` adds 5/15/30/60 series for charting and indicators only.
- `BarsInProgress` map in the template: `0` = primary 1m (signals and management), `1..4` = optional advisory 5/15/30/60, last = optional 1-tick management series.
- `Calculate`: the template uses **`Calculate.OnBarClose`**, matching the Python 1m-close loop. BUILD_PLAN §8 recommends `Calculate.OnPriceChange` (or Tick Replay in backtests) so a far-side break fires on the first trade through the level. **TODO:** decide per plugin. It changes fills (§6.2) and must be recorded in the run notes.

3.3 Sessions (ET)

Session use	Value	Source
Entry gate (host default)	09:30 ≤ t < 11:00 ET (`AllowOpenWindow=true`)	`default.yaml sessions`
Entry gate (Nitro mapping)	09:30 ≤ t < 12:00 ET	`params.essentials_mapping`
Contract `session` hint enum	`RTH, ETH, RTH_POST_IB, RTH_OPEN_AVOID, ANY_LIQUID` ("e.g. CME RTH 09:30–16:00 ET")	contract §3
EOD flatten	first bar ≥ 16:00 ET, or ET day change	`backtest_m1.py` (harness, **not** Essentials)
Day key for caps	ET calendar date	`engine._day_key`
CME session day (18:00 ET)	**not used**: `one_trade_reset_on_session` is passthrough	**GAP G7**
Named sessions (PreMarket, Lunch, Close, After, Asia/London/USA), Friday close 22:15, daily close 22:15 / reopen 01:15, `eod_exit_time 06:58:30`	stored only	passthrough. The SetupHunter NT8 code applies them in **PC-local** time (`Time[0].TimeOfDay`): **GAP** (zone undefined for the farm)

NT8 session template: use **CME US Index Futures ETH** (or the instrument's default ETH template) so `Time[0]` covers the whole electronic day. The chassis does its own ET gating; never rely on `IsExitOnSessionCloseStrategy` (set to `false` in the template).

4. Plugin interface: methods, inputs, outputs, lifecycle

4.1 Python (reference)

- `essentials_host.plugin.EntryPlugin` (Protocol): `emit(*args, **kwargs) -> dict` returns **one** ENTRY contract dict (schema 1.0). This is the only required method.
- Streaming convention (nitro_crt_v2): `on_bar(ts, o, h, l, c) -> list[dict]`, where `ts` is the ET-aware **bar start**, returning 0..n contracts that the caller passes to `Essentials.accept_signal(sig, ts=bar_ts)`.
- Adapters: `signal_from_nitro_crt(...)` (wraps the v1 `build_signal`), `signal_from_smt_advanced(...)` (minimal dict; `source_lane` defaults to `other`).

4.2 NT8 template: `IEntryPlugin` (`SMT_AllinOne_Roman_Contracts.cs`)

Member	Direction	Purpose
`string PluginId, PluginVersion, SourceLane`	out	Identity, written to logs (`nitro_crt_v2`, `crt_v2_m1`, `nitro_crt`)
`int[] RequiredTimeframesMinutes`	out	Clock TFs the chassis must build (each divides 60)
`void Initialize(PluginContext ctx)`	in	Called once in `State.DataLoaded`. `ctx` = `SymbolRoot`, `TickSize`, `PointValue` (from `MasterInstrument`), `EtZone`, `Log`. **Throw** `NotSupportedException` when a param switches on an unbuilt feature (Python raises `NotImplementedError`: no silent no-ops)
`List<EntrySignal> OnBar(Bar1m bar, List<HtfBar> closedBefore, List<HtfBar> closedWith)`	in → out	Called once per primary 1m bar close, in time order. `bar.StartEt` is the ET bar start. Returns 0..n signals
`void OnSignalResult(EntrySignal s, bool accepted, string code, string reason)`	in	Template addition (Python has no callback). Optional bookkeeping only
`void OnTradeClosed(ClosedTradeInfo t)`	in	Template addition. **Must not be used to build trade caps**

4.3 Lifecycle (NT8 `OnStateChange` → events)

NT8 state / event	Chassis does	Plugin sees
`State.SetDefaults`	Strategy settings (`Calculate.OnBarClose`, `EntriesPerDirection=1`, `EntryHandling.AllEntries`, `IsExitOnSessionCloseStrategy=false`, `StopTargetHandling.PerEntryExecution`, `StartBehavior.WaitUntilFlat`, `TimeInForce.Gtc`, `RealtimeErrorHandling.StopCancelClose`, `IsInstantiatedOnEachOptimizationIteration=true`). Essentials defaults from `default.yaml`, then `SetPluginDefaults()` (plugin intents override)	—
`State.Configure`	`Slippage` from `UseSlippage`; optional `AddDataSeries` (advisory 5/15/30/60, tick 1)	—
`State.DataLoaded`	Resolve ET and chart zones; check the 1m primary; read `MasterInstrument` and assert the spec table; warn on stub params; `CreatePlugin()` → `Initialize(ctx)`; build `ClockAggregators`; open log files under `Documents\NinjaTrader 8\SMT_AllinOne_Roman\`	`Initialize`
`OnBarUpdate`, `BarsInProgress==0`	(1) ET conversion; (2) EOD flatten check; (3) BE/trail on the open trade; (4)	`OnBar`

NT8 state / event	Chassis does	Plugin sees
	update aggregators; (5) `plugin.OnBar`; (6) `AcceptSignal` for each signal	
`OnBarUpdate`, tick BIP	BE/trail with the tick price (optional)	—
`OnOrderUpdate`	Unfilled 1-bar Stop/Limit entry `Cancelled`/`Rejected` → trade voided (day count kept)	—
`OnExecutionUpdate`	Entry fill → record the fill (avg for partials), optional bracket shift. Exit fill (`FromEntrySignal == trade`) → exit reason, close when fully exited, write trades CSV	`OnTradeClosed`
`State.Terminated`	(nothing yet) **TODO:** flush and summary file	—

5. Entry signal contract (schema 1.0)

Source: `ENTRY_SIGNAL_CONTRACT.md`. C# mirror: `EntrySignal` (+ `ProfileRefs`, `ConfluenceLineup`, `LineupMember`). One signal object per confirmed trigger event. A Confluence Lineup emits **one** merged signal.

5.1 Fields

Field	JSON type	C# member	Contract	Essentials v0.1 host (`_validate_signal`)	OM use
`schema_version`	string `"1.0"`	`SchemaVersion`	required	not checked	ignore (logging)
`signal_id`	string	`SignalId`	required; UUID or `{setup_id}-{symbol}-{yyyyMMddHHmmss}`	not checked; used as `OpenPlan.signal_id` (else `trade_id`)	idempotency key. **GAP:** no dedupe in v0.1
`emitted_at`	string ISO-8601 + offset	`EmittedAtIso` / `EmittedAtEt`	required	used as the session/day clock when `accept_signal(ts=None)`; if missing **and** no `ts`, `datetime.now()` is used (**GAP**)	audit
`setup_id`	string	`SetupId`	required (stable pack id)	**required**	routing / tags
`source_lane`	string enum	`SourceLane`	required: `volume_profile`, `nitro_crt`, `lanto`, `dezi`, `riley`, `other`	**required** (value not checked)	catalogue. **TODO:** new traders (ICT2023, ORB, Nitro1/2, Dhesi…) have no lane value. Use `other` or extend the enum via a contract change
`symbol`	string	`Symbol`	required: NQ, ES, YM, GC, CL, MNQ, MES, MGC, …	**required**	order instrument. **GAP G1:** host `on_bar` has no symbol check
`direction`	string	`Direction`	required: `long` / `short`	**required**, must be long/short	order side
`trigger_price`	number	`TriggerPrice`	required, > 0	**required**, > 0	entry reference (Python fills here)
`stop_hint`	number	`StopHint` (double?)	**required**	**optional**: baseline `entry ∓ baseline_stop_loss_points`	hint; OM may adjust
`target_hint`	number or number[]	`TargetHint` (`List<double>`)	**required**, T1 first	**optional**: baseline `entry ± baseline_profit_target_points` (or Roman ladder)	hint; v0.1 manages **T1 only**
`session`	string enum	`Session`	required: `RTH`, `ETH`, `RTH_POST_IB`, `RTH_OPEN_AVOID`, `ANY_LIQUID`	**required** (value not used)	**GAP:** not mapped to a clock window; the host uses only its own open window
`grade`	string enum	`Grade`	required: `A+`, `A`, `A-`, `B+`, `B`, `C`	**required** (value not used)	**GAP:** no size tier or minimum-grade gate in v0.1
`timeframe_context`	string	`TimeframeContext`	recommended	—	tag
`timeframe_trigger`	string	`TimeframeTrigger`	recommended	—	tag
`profile_refs`	object	`ProfileRefs`	optional (VP lanes): `prior_vah`, `prior_val`, `prior_poc`, `d_poc`, `session_vah`, `session_val`, `hvn`, `lvn`,	—	audit

Field	JSON type	C# member	Contract	Essentials v0.1 host (_validate_signal)	OM use
			npoc, ibh, ibl, profile_shape		
meta	object	Meta	optional, free-form	ignored (**GAP G2:** meta.om_hints not read)	ignore
confluence_lineup	object or null	ConfluenceLineup	optional: is_lineup, window_seconds (300), direction, members[≥2] {setup_id, source_lane, grade, signal_id, trigger_price, emitted_at}, lineup_grade, primary_setup_id	copied to OpenPlan.confluence_lineup only	**GAP:** no lineup router/merger exists

Strict geometry (nitro_crt_v2 validate_signal, C# ValidateStrict): long stop < trigger < T1; short stop > trigger > T1. **Forbidden keys:** quantity, size, contracts, trail, max_trades.

Validation levels in the NT8 template: StrictContractValidation=false (default) = host checks only (Python parity). true = full contract. Plugins should call ValidateStrict() themselves before returning a signal (Nitro does).

5.2 Example JSON (Nitro CRT v2 M1, shape from build_signal_v2 / plugin.emit; values illustrative)


```
{
  "schema_version": "1.0",
  "signal_id": "NITRO_CRT_V2_BREAK-MNQ-20260310103700",
  "emitted_at": "2026-03-10T10:37:00-04:00",
  "setup_id": "NITRO_CRT_V2_BREAK",
  "source_lane": "nitro_crt",
  "symbol": "MNQ",
  "direction": "short",
  "trigger_price": 24510.25,
  "stop_hint": 24530.75,
  "target_hint": [24479.5],
  "session": "RTH",
  "grade": "B",
  "timeframe_context": "15m",
  "timeframe_trigger": "1m",
  "meta": {
    "plugin_version": "crt_v2_m1", "crt_tf_min": 15,
    "c1_start": "2026-03-10T10:00:00-04:00", "c2_start": "2026-03-10T10:15:00-04:00",
    "formed_at": "2026-03-10T10:30:00-04:00",
    "c1_high": 24540.0, "c1_low": 24479.5, "c2_high": 24545.25, "c2_low": 24510.5, "c2_close": 24520.0,
    "sweep_extreme": 24545.25, "entry_mode": "break", "stop_mode": "target_rr", "stop_rr": 1.5,
    "contract_spec": {"point_value": 2.0, "tick_size": 0.25},
    "data_proxy_root": "NQ", "bias": "not_evaluated_m1", "one_crt_per_day": true
  },
  "confluence_lineup": null
}
```


(Stop check: |T1 − fill| = 30.75 / 1.5 = 20.5, so the stop is 24510.25 + 20.5 = 24530.75, rounded up to a tick, away from entry.) The contract also has §6/§7 examples for a VP solo signal and a Confluence Lineup.

6. Order and position handling

6.1 Accept / reject (Python `accept_signal` = C# `AcceptSignal`, same order)

Step	Check	Reject code	Notes
1	Validate (host or strict)	`invalid_signal`	missing fields / bad direction / trigger ≤ 0
2	Session gate: `open_window_start ≤ t < open_window_end` in ET (skipped if `allow_open_window=false`)	`session_reject`	clock = `ts` arg, else `emitted_at`. The NT8 template uses the **bar start ET** (harness parity: `accept_signal(sig, ts=bar_ts)`)
3	`len(open_positions) ≥ max_open_trades`	`max_open_trades`	`max_open_trades=0` rejects everything. The NT8 template supports **0 or 1** (`Active` single slot); see GAP
4	Day limit: `use_one_trade_mode` → `max(1, one_trade_max)`; else `max_trades_per_day>0`; else `day_trade_cap>0`; else none	`day_trade_cap`	counted per **ET date**, incremented **at accept**
5	Resolve stop/targets (hints win, else baselines), size, build `OpenPlan`	—	`trade_id = EH-{seq:04d}-{uuid8}`

Rejects are returned as `Reject(reason, signal_id, code)`. The NT8 template logs them to `*_rejects.csv` and calls `plugin.OnSignalResult`.

6.2 Fills

	Python Essentials v0.1	NT8 template
Entry	**Instant, at `trigger_price`**, on the signal bar; no order object (**GAP G6**)	Managed order, chosen by `EntryFillMode`: `MarketNextBar` (`EnterLong`/`EnterShort`; fills at next bar open with `OnBarClose`), `StopAtTrigger` (`EnterLongStopMarket(0,false,qty,trigger,name)`, 1-bar life), `LimitAtTrigger` (`EnterLongLimit`, 1-bar life). **None is an exact twin**; parity must be measured (TODO)
Gap fill vs plan	stop/target stay at the hint prices	`BracketOnFill = KeepHintPrices` (Python parity) or `ShiftBracketToFill` (SetupHunter v6.8.3 FILL_FIX: shift SL/TP by `fill − planned`)
Order names	—	entry `AR_L_<seq>` / `AR_S_<seq>`; NT bracket `Stop loss` / `Profit target`; EOD `AR_EOD`
Slippage / commission	not applied (`money_mgmt.use_slippage`, `commission_equity` unwired)	`UseSlippage` → `Strategy.Slippage` (ticks), as in SetupHunter. Commission via the NT commission template (**TODO:** set it per account; `extra_commission_per_side` not ported)
Unfilled entry	n/a	`OnOrderUpdate` Cancelled/Rejected → trade voided, **day count kept** (accept-time parity)
Partial fills	n/a	entry average price; trade closes when exited qty ≥ filled qty

Nitro F4 fill rule inside the plugin: `fill = max(open, entry_level)` for longs and `min(open, entry_level)` for shorts (gap-through fills at the open).

6.3 Stop / target

- Stop = `stop_hint`, else `entry ∓ baseline_stop_loss_points` (default 300 **price points**). Targets = `target_hint` list, else `entry ± baseline_profit_target_points` (1500). When `exit_mode ≠ OneShot` and there is no hint, the Roman ladder `roman_target_pts [50,150,400,800]` is used.
- Only **T1** is managed (`_hit_target` uses `targets[0]`). Multi-leg is plan-only (`LegPlan`), and `roman_multi_leg_live_exits = False` (**GAP**).
- NT8: `SetStopLoss(name, CalculationMode.Price, RoundToTickSize(stop), false)` and `SetProfitTarget(name, CalculationMode.Price, T1)` are called **before** the entry order (managed-order rule). Python does not round baseline prices to tick; NT must (`MasterInstrument.RoundToTickSize`).

6.4 Break-even (`_apply_break_even`)

`if use_break_even and not be_triggered and MFE ≥ trigger` then `stop = entry ± buffer` (tighten only) and `be_triggered = True`. Here `trigger = break_even_trigger_points` (100; falls back to `roman_be_trigger_pts[0]` if None), and `buffer = break_even_buffer_points` (2; if 0, it uses `roman_be_lock_pts` = 1). This is **one fixed-point trigger for every signal** (**GAP G2:** no per-signal or "near target and failing" BE). The NT8 template applies the same logic and moves the stop with `SetStopLoss(..., Price, newStop, false)`, as in `SMT_Advanced.ManageBreakeven`.

The Ratchet BE ladder (`Ratchet_BE1..3`) and `Roman_BE_TriggerPts1..4` exist in the TDL / SetupHunter NT8 code but are **not** used by Essentials v0.1 (passthrough).

6.5 Trailing (`_apply_trail`)

`if use_staged_trail`, pick the **highest** stage `i ≥ current` with `MFE ≥ trigger_i`; then `stop = (entry ± MFE) ∓ dist_i`, tighten only. Defaults: (70, 250), (150, 250), (250, 350) price points; off by default. Same in NT8 (`Trail1..3_TriggerPts/DistPts`, SetupHunter property names). Frequency jump, structure trail, per-leg trail and ATR trail are **not** in Essentials v0.1.

6.6 EOD flatten

Essentials v0.1 has no public flatten. `ExitReason` includes `"eod"`, `eod_exit_time` (06:58:30, DNA sample) is passthrough, and the M1 harness closes at the first bar ≥ 16:00 ET (or on a new ET day) through the private `Essentials._close()` at the last bar's close (4 of 187 trades) (**GAP G3**). NT8 template: `UseEodFlatten=true`, 16:00 ET. `ExitLong/ExitShort("AR_EOD", entrySignal)` fills at the next bar open, which differs from Python's last-close fill.

6.7 Ties and ordering inside a bar

- Python order per bar: MFE update → BE → trail → **stop check before target** ("conservative: stop before target if both touch same bar"). The BE/trail stop moved on a bar is tested against **that same bar's** low/high. Exit reason: `trail_stop` if a trail stage is active, else `be_stop` if BE fired, else `stop`; `target` otherwise.
- Harness entry-bar policies (`backtest_m1.py`): **conservative** runs the entry bar's full range through `on_bar` (stop first); **next_bar** ignores the rest of the entry bar. Both must be reported (§10).
- NT8: stop/target fills are done by NT's fill engine, which with 1m bars and `OrderFillResolution.Standard` does not know the intrabar path. A stop moved on bar *n* is live from bar *n+1* (or the next tick with `UseTickManagementSeries`). Use `OrderFillResolution.High` (1 tick) or Tick Replay for ambiguity studies. **TODO:** measure NT-vs-Python divergence on the same data.

7. Risk engine

Control	Python v0.1 status	Chassis param (NT8)	Default (host)	Notes
Fixed size	**wired**	`FixedQuantity`	1	`_size_quantity`
Roman lots (sum)	wired as a sum when `exit_mode ≠ OneShot`	`Roman_Lot1..4`	1/1/1/0	legs are plan-only
Risk-% sizing	**STUB** (`risk_percent_sizing.wired=False`)	`UseMoneyManagement`, `RiskPercent`, `VirtualStartingBalance`	off / 1.0 / 10000	SetupHunter NT8 has `ComputeQuantity = floor(equity × risk% / (stopDist × PointValue))`. **Not** ported: needs David's go-ahead
MM simulator / martingale (056)	passthrough	—	off	
Max open trades	**wired**	`Advanced_MaxOpenTrades`	1	template: 0/1 only
One-trade mode	**wired** (per ET day; global across symbols when one host instance is shared)	`UseOneTradeMode`, `OneTradeMax`	off / 2	Nitro maps true / 1
Max trades per day	**wired**	`Advanced_MaxTradesPerDay` (`shutoffs.max_trades_per_day`)	0 (off)	
Day trade cap	**wired**	`DayTradeCap` (`shutoffs.day_trade_cap`)	10 (DNA sample)	the active cap with host defaults
Session max trades / session caps per block	seeded into the Advanced lane only / passthrough	—	0 / {premarket 5, …}	not in `engine.py`
`one_trade_reset_on_session` (CME 18:00 day)	passthrough	—	true	**GAP G7**
`one_shot_per_direction`, `duplicate_tolerance_points`, `divide_longs_shorts`, `max_trades_per_setup`	passthrough	—		
Session gate	**wired** (open window only)	`AllowOpenWindow`, `OpenWindow*`	09:30–11:00 ET	
Named sessions, Friday/daily close	passthrough	—		§3.3
News guard	**scaffold passthrough** (`Advanced_UseNewsGuard`, `NewsMinutesBefore/After=30`, `NewsManualTimesCSV`)	stored in "99. Passthrough"; a warning is printed if on	off	**GAP G5:** Nitro does FOMC/no-news-Monday plugin-side; CPI/NFP list not sourced
Prop rules (daily loss %, profit lock %)	**scaffold passthrough** (`Advanced_UsePropFirmGuard`, `MaxDailyLossPercent`, `DailyProfitLockPercent`)	stored only	off	SetupHunter NT8 has `UseDailyLossLimit/ProfitTarget/EquityProtection` (currency); **not** in Essentials. Prop consistency is listed as OM-owned in the contract: **GAP**
Tick shutoffs (051/052)	passthrough; group 051 rows have **no property names** in the bag (`051_Tick_Count_Filters_Overall_Shut_Off = —`)	—		**GAP:** bag data quality
HTF bias gate	seeded into the Advanced lane (`UseHTFBiasGate`, `HTFMinutes`)	—	off	not used by `engine.py`

Control	Python v0.1 status	Chassis param (NT8)	Default (host)	Notes
Grade → size tier / minimum grade	**not implemented**	—		**GAP**

8. Instrument specs

Symbol	Exchange	Point value (USD/pt)	Tick	Tick value	Data proxy	Source
MNQ	CME	2.00	0.25	$0.50	NQ	`specs.py` (CME Micro E-mini FAQ, verified 2026-10-01)
MES	CME	5.00	0.25	$1.25	ES	same
MGC	COMEX	10.00	0.10	$1.00	GC	CME/CFTC Micro Gold rulebook ch. 120
NQ	CME	20.00	0.25	$5.00	NQ	`specs.py`
ES	CME	50.00	0.25	$12.50	ES	`specs.py`
GC	COMEX	100.00	0.10	$10.00	GC	`specs.py`
Stocks	—	**GAP/TODO**	**GAP/TODO**	—	—	not in `specs.py`; read `MasterInstrument` and add a verified row

- NT8: the chassis reads `Instrument.MasterInstrument.Name / TickSize / PointValue` in `DataLoaded` and **disables itself** on a mismatch with the C# table (`InstrumentSpecs`), or when the root is missing and `RequireVerifiedSpec=true`.
- **GAP G4:** Essentials PnL is `points × qty` with no point value. USD = points × `point_value` comes from the plugin (`specs.py`) or the harness. The NT8 template writes `pnl_usd_gross = pnl_points × PointValue`.
- **Units warning:** Essentials Python and this template use **price points** (300 = 300 NQ points). The SetupHunter v6.8.3 NT8 code (the source of TDL "wired" statuses) multiplies its "points" by `StrategyPointSize = 0.01` (MT5 points), so the same TDL value 300 means **3.00 price units** there. **GAP:** never copy TDL point values between the two without converting.
- Micro data: NQ/ES/GC prices are used for MNQ/MES/MGC. Volume-based logic on micros must not use full-size volume without saying so (contract §8.5 prefers exchange volume).

9. Param categories and groups

Full list (294 rows, 232 unique properties): `smt-python/docs/SMT_ALLINONE_ROMAN_PARAMS.md`. Machine bag: `config/essentials_host/essentials_params_full.json` (and `.yaml`). Group index: `config/essentials_host/essentials_groups.json`. Source: Essentials CATEGORIES allowlist × TDL `zz-strategy-params.json` (SetupHunter v6.8.3, 941 params / 117 groups in total).

Counts: **11 categories, 37 groups, 294 params**; `tdl_wired_like` 175; `python_twin_mapped` 21; **`python_pnl_wired_or_seeded` 14** (only these change Python backtest PnL). Everything else is passthrough.

Per-category totals: sessions 61 · trend-filters 60 · risk-entry 35 · scale-in 32 · exits 26 · host-toggles 23 · shutoffs 21 · money-mgmt 16 · frequency-jump 10 · lots 5 · commission-equity 5.

Column meaning: *Python PnL wired/seeded* = affects Python sim. *TDL/Ninja wired-like* = wired in the SetupHunter NT8 code but passthrough in Python. *Pure passthrough* = TDL scaffold.

Category	#	Group	Params	Python PnL wired/seeded	TDL/Ninja wired-like, Python passthrough	Pure passthrough (scaffold)	Wired/seeded properties
host-toggles	14	003. Trade Filters	23	2	4	17	OneTradeMax, UseOneTradeMode
lots	17	004. Lot Settings	5	0	4	1	—
risk-entry	18	006. Entry, Stop and Target	35	5	20	10	BaselineProfitTargetPoints, BaselineStopLossPoints, CustomProfitTargetPoints, CustomStopLossPoints, Roman_SLPts
exits	19	007. BreakEven	17	2	15	0	Advanced_BreakEvenTriggerPoints, Advanced_UseBreakEven
exits	20	008. Trailing Stop	9	0	8	1	—
frequency-jump	21	009. Frequency Jump	10	0	10	0	—
scale-in	22	010. Scale In - Entry, Stop and Target	12	0	12	0	—
scale-in	23	011. Scale In - BreakEven	8	0	8	0	—
scale-in	24	012. Scale In - Trailing Stop	12	0	12	0	—
trend-filters	53	027. Trend Filters - MAs	6	0	0	6	—
trend-filters	54	028. Trend Filters - Custom Trail Two	6	0	0	6	—
trend-filters	55	029. Trend Filters - VWAP	1	0	0	1	—
trend-filters	56	030. Trend Filters - Volume Width Bars	16	0	0	16	—
trend-filters	57	031. Trend Filters - Caught Orders	2	0	0	2	—
trend-filters	58	032. Trend Filters - Swings	9	0	0	9	—
trend-filters	59	033. Trend Filters - VolReader	6	0	0	6	—
trend-filters	60	034. Trend Filters - Market Bid Ask Delta	7	0	0	7	—
trend-filters	61	035. Trend Filters - VolPress	4	0	0	4	—
trend-filters	62	036. Trend Filters - Timeframes Bias	3	2	1	0	HTFMinutes, UseHTFBiasGate
sessions	101	042. Trading Times Calendar	13	0	13	0	—
sessions	102	043. Time Sessions	1	0	1	0	—

Category	#	Group	Params	Python PnL wired/seeded	TDL/Ninja wired-like, Python passthrough	Pure passthrough (scaffold)	Wired/seeded properties
sessions	103	044. Time Sessions - 1. PreMarket	5	0	5	0	—
sessions	104	045. Time Sessions - 2. Market Open	5	1	4	0	AllowOpenWindow
sessions	105	046. Time Sessions - 3. Lunch	5	0	5	0	—
sessions	106	047. Time Sessions - 4. Market Close	5	0	5	0	—
sessions	107	048. Time Sessions - 5. After Market	11	0	0	11	—
sessions	108	049. Time Sessions - Styles	5	0	0	5	—
sessions	109	050. Time Sessions - Global World Time Sessions	11	0	0	11	—
shutoffs	68	051. Tick Count Filters - Overall Shut Off	5	0	0	5	—
shutoffs	69	052. Tick Count Filters - Sessions Shut Off	1	0	1	0	—
shutoffs	72	053. Trades Count Filters - Number of Trades	2	1	0	1	SessionMaxTrades
shutoffs	71	054. Session Exit Filters - Sessions Trades	6	1	5	0	AllowOpenWindow
shutoffs	70	055. Day Time Exit Filters - End Of CME Session	7	0	7	0	—
money-mgmt	99	056. Money Management Simulator	13	0	13	0	—
money-mgmt	100	057. Slippage Management	3	0	3	0	—
commission-equity	112	098. Additional comissions	2	0	2	0	—
commission-equity	113	099. Equity Curve Analyzer	3	0	3	0	—

NT8 template property groups (`GroupName` follows the TDL group names so the PC2 AllinOne/TDL layout can be matched later): `00. Chassis / Data`, `003. Trade Filters`, `004. Lot Settings`, `006. Entry, Stop and Target`, `007. BreakEven`, `008. Trailing Stop`, `045. Time Sessions - 2. Market Open`, `055. Day Time Exit - EOD`, `057. Slippage Management`, `99. Passthrough (unwired v0.1)`, plus `10. Plugin - <Trader>` on each concrete strategy. Only the PnL-relevant subset is exposed. The other ~250 properties stay in the JSON bag until a feature is wired (**TODO:** generate the remaining properties from the bag only when they get behaviour; do not add dead knobs).

10. Backtest harness and reporting standard

10.1 Harnesses that exist

- **AllinOne power surface:** `python -m smt_strategy.run_smt_allinone_roman --preset <name>` (alias `run_allinone`; `--list`, `--dry-merge`, `--grid`). Output `output/smt_allinone_roman/<preset>/`: `trades.csv`, `summary.csv`, `summary.json` (with `school_gate_ok`, `school_gate_note`, `essentials_unwired`), `equity.png/.csv`, and playbook extras. Power matrix: `scripts/power_smt_allinone_roman.py` → `LEADERBOARD.md`.
- **Plugin harness (pattern for every trader):** `entry_plugins/nitro_crt_v2/python/backtest_m1.py`. Loop per ET 1m bar: (1) `Essentials.on_bar` for earlier positions; (2) `plugin.on_bar` → `accept_signal(sig, ts=bar_ts)`; (3) entry-bar policy; (4) EOD flatten 16:00 ET. Outputs `summary.json`, `trades.csv`, `signals.jsonl`, `events.csv` (plugin funnel).
- **NT8:** Strategy Analyzer on a Dev PC, plus the chassis logs (`*_signals.jsonl`, `*_trades.csv`, `*_rejects.csv`). **Deferred** until David approves (BUILD_PLAN §8).
- Research and backtests run on **PC2 / PC3** (`C:\GROK_BOT_TradingFarm`). The box is for code and unit tests.

10.2 Metrics (every run, per symbol and combined)

From `backtest_m1.metrics` and the BUILD_PLAN test plan: trades, wins, losses, **win rate**, **profit factor**, gross profit / loss $, **net $** for 1 contract (state which: e.g. 1 MNQ), **max drawdown $** (closed-trade equity), **avg R** (R = |entry − initial stop|), exits by reason (`stop/target/be_stop/trail_stop/eod`), results by TF, by weekday and by setup, the plugin **funnel** (formed / selected / triggered / expired / cancelled / ambiguous / rejected by OM with code), data range in ET, RTH days, data file hashes, machine, code version, and params JSON. The AllinOne leaderboard `score_row` is a 0–1 blend (0.35 gate + 0.25 PF + 0.20 PnL + 0.10 trades + 0.10 WR).

10.3 Same-minute ambiguity, reported both ways (mandatory)

1m OHLC cannot tell whether stop or target (or stop vs entry break) came first. Report **both**: * **Conservative** (default headline): stop before target, and the entry bar's full range is checked. * **Next-bar / optimistic:** ignore the rest of the entry bar. Nitro M1 example: 66 of 187 trades exited on their entry minute (61 stops). PF 0.776 conservative vs 0.881 next-bar, about $1,000 apart. The fix is sub-minute data (missing) or NT Tick Replay.

10.4 Costs

Report **$0 cost** and an **assumed round-turn cost** labelled as an assumption (M1 used $1.50 RT per MNQ: "an assumption, not a broker quote"). Slippage: Python unwired. NT8 uses `Strategy.Slippage` ticks plus the commission template. **TODO:** agree farm cost assumptions per instrument.

10.5 Hygiene

No tuning on the full sample: choose params in-sample (first ~2/3), report out-of-sample (last ~1/3) (Nitro M4). Every result says "incomplete milestone, not a verdict" until all features are built.

11. Validation, school gate and promotion flow

11.1 Tests

- Unit (box, pytest): specs, params lock, ET conversion and roll exclusion, aggregator parity and no look-ahead (across DST and gaps), geometry, tie-breaks, contract validation, end to end through the real `Essentials`. Nitro M1 has 29 tests (`entry_plugins/nitro_crt_v2/tests`); the host has `tests/test_essentials_host.py` and `tests/test_essentials_runtime.py`.
- Video replay: the plugin reproduces the trader's examples (direction, TF, approximate levels) and skips the examples the rules exclude.

11.2 School gate

- **VALID** only with **≥ ~3 months of data (code: `SCHOOL_PREFER_DAYS = 90`) AND ≥ 100 trades** per reported configuration (`allinone/runner.school_gate_note`). Otherwise the result is `EXPLORATORY` / "insufficient sample".
- Passing the gate means the sample is big enough to read. It is **not** a verdict.

11.3 Promotion flow

1. Python plugin, milestone by milestone, on the box (code + tests).
2. Backtests on PC2/PC3 with the school gate, both ambiguity policies and both cost levels.
3. **Pin rule:** results that score **70+** (on a 0–100 scale; the power `score_row` is 0–1, so 0.70) **or grade A** are pinned to the **Dev keeps board**. **TODO:** the board's location and pin mechanism were not found in the code read for this template.
4. **Dev only.** No production, no live website publish (parity doc: "Site / VPS publish — out of scope").
5. **Ninja port later:** compile this NT8 template on a Dev PC → parity fixture (Python `signals.jsonl` vs NT log: identical `setup_id`/direction/trigger/stop/target per day on the same data) → Strategy Analyzer with both fill resolutions → Sim101 → live only with David's explicit approval.

11.4 NT8 template build status

- Written on the box. **Not compiled in NT8; not run.**
- **Box syntax check (2026-10-01):** all six `.cs` files compiled cleanly (0 errors, 0 warnings) with the .NET 8 SDK at **C# language version 5**, against hand-written **stubs** of the NT8 API members they use. This catches syntax and type errors only; it does not prove the real NinjaTrader assemblies match (for example exact overloads or `MasterInstrument` members).
- First real compile: NinjaScript Editor on a Dev PC, after David's go-ahead.

12. Known gaps in v0.1

Gaps G1–G7 come from `nitro_crt_v2/BUILD_PLAN.md` §4. C1–C16 were found while writing this template. **Each one needs David's go-ahead before anyone changes Essentials.**

ID	Gap	Where	Impact / workaround
G1	`on_bar()` applies one bar to **all** open positions; no `symbol` check	`engine.on_bar`	Multi-symbol needs per-symbol routing. Workaround: feed only the open symbol's bars (`max_open_trades=1`). NT8: one strategy instance per instrument, so N instances do **not** share caps (**GAP** for one-trade-total across symbols)
G2	BE is one global fixed-point trigger; `meta` / `meta.om_hints` ignored	`_apply_break_even`	Per-signal BE ("near TP and failing") is impossible
G3	No public flatten / EOD exit; `eod_exit_time` passthrough	engine	Harness calls the private `_close()` at 16:00 ET. NT8 template adds `UseEodFlatten` (harness parity)
G4	No instrument spec table in the host; PnL is points × qty; risk-% sizing stub	engine	USD from plugin `specs.py`; the NT8 template uses `MasterInstrument.PointValue`
G5	News guard params are scaffold passthrough	294 bag	Plugin-side calendar filters for now
G6	Instant fill at `trigger_price`; no slippage/commission; same-bar stop-before-target	engine	Report both ambiguity policies + cost levels
G7	`one_trade_reset_on_session` passthrough; caps reset on the ET calendar day, not CME 18:00	`_day_key`	Matters for Asia/London setups
C1	Naive datetimes treated as **UTC** in `_as_dt`; missing ts falls back to `datetime.now()`	engine	Silent session shift; always pass tz-aware ET
C2	Contract `session` and `grade` are required but **unused** (no clock map, no size tier, no min grade)	engine	Grade affects nothing in OM yet
C3	Contract requires `stop_hint`/`target_hint`; the host treats them as optional and fills 300/1500-point baselines	engine vs contract	A plugin bug silently becomes a 300-point stop; use strict validation in plugins
C4	Only T1 managed; Roman multi-leg lots/targets/BE per leg plan-only	`_build_legs`	`ExitMode=RomanMulti` sums lots onto one stop/target
C5	No `signal_id` dedupe / idempotency	engine	Duplicate emits could double-count caps
C6	No Confluence Lineup router/merger, and no opposing-direction policy	—	Contract §5 is unimplemented
C7	`source_lane` enum lacks most farm traders (ICT2023, ORB, Nitro1/2, Dhesi, Chad, …)	contract §3	Use `other` or extend the contract (versioned)
C8	Day cap counted at **accept**; NT unfilled Stop/Limit entries still consume the cap	engine / template	Policy decision needed
C9	BE/trail move the stop and test it on the **same bar** in Python; in NT the moved stop is live next bar/tick	engine vs NT	Parity divergence; measure
C10	**Units:** Essentials uses price points; SetupHunter NT8 (TDL "wired" source) uses MT5 points × `StrategyPointSize` (0.01)	TDL vs Python	The same "300" means different stops

ID	Gap	Where	Impact / workaround
C11	Prop-firm guard, daily loss / profit lock: scaffold only; SetupHunter has currency-based versions not in Essentials	294 bag	No prop protection in the chassis
C12	Group 051 (Tick Count Overall Shut Off) rows have no property names in the bag	`essentials_params_full.json`	Bag data quality
C13	No CME holiday / early-close calendar; no CPI/NFP list	data	Holiday sessions traded
C14	No roll-date exclusion in the NT8 chassis; stock instrument specs missing; stock trade-print builder path not located	data / template	Post-filter rolls; add verified stock specs
C15	`max_open_trades > 1` not supported by the NT8 template (single `Active` slot); NT managed `EntriesPerDirection=1`	template	Fine for v0.1 presets (all 1)
C16	Dev keeps board location/pin mechanism not found; school-gate “score 70+” scale vs `score_row` 0–1	process	Confirm the board and scale

13. FILL-IN TRADER SECTION (copy once per trader)

Instructions: fill every cell. Unknown → `TODO`. Anything the chassis cannot do → add to §12 and to the "Essentials mapping and gaps" table. Never implement OM inside the plugin. Machine copy: `trader_plugin_template.yaml`.

13.1 Identity

Field	Value
Trader name	
Setup name(s) / `setup_id`(s)	
`source_lane`	
Plugin folder (Python)	`entry_plugins/<trader>_<setup>_v<N>/`
NT8 strategy / plugin class	`SMT_AR_<Trader><Setup>_v<N>` / `<Trader><Setup>Plugin`
Plugin version / milestone	
Owner / written (Brisbane)	David Oliver /
Instruments	
Status	draft / M1 / … / validated

13.2 Source videos

#	Title	URL / video id	Upload date	Analysis folder	Key timestamps
1					
2					

13.3 Setup rules (boolean, testable)

#	Rule	Evidence (video @ mm:ss)	Feature id	Param(s)
R1	Context / bias			
R2	Structure			
R3	Trigger			
R4	Invalidation → `stop_hint`			
R5	Target(s) → `target_hint`			
R6	Filters (time, news, chop, …)			
R7	Grade ladder (A+/A/B…)			

#	Rule	Evidence (video @ mm:ss)	Feature id	Param(s)
R8	Setup-eligibility (e.g. first per day)			

13.4 Params with defaults

Param	Default	Type / allowed	Feature	Built?	Notes

13.5 Feature → milestone map

#	Feature	Video evidence	Milestone	Module (Python)	C# location	Status
F1			M1			

13.6 Essentials mapping and gaps

Intent	Existing chassis param	Mapped value	Wired in Python?	Wired in NT8 template?
Entry session window	`AllowOpenWindow`, `OpenWindow*`		✅	✅
Trade cap	`UseOneTradeMode/OneTradeMax` or `Advanced_MaxTradesPerDay`		✅	✅
No overlap	`Advanced_MaxOpenTrades`		✅	✅
Size	`FixedQuantity`		✅	✅
BE / trail	`Advanced_UseBreakEven`, `UseStagedTrail`		✅ (fixed pts)	✅
News	`Advanced_UseNewsGuard`		❌	❌

13.7 Data needs and status

Need	Status	Detail (path, range ET, clock, rolls excluded, hash)

13.8 Run record

Run	Machine	Data	Ambiguity policy	Cost	Trades	WR	PF	Net $	Max DD $	School gate

13-EX. WORKED EXAMPLE: Nitro Trades CRT v2 (filled)

Snapshot: `crt_v2_m1`, as read on 2026-10-01 at about 14:20 Brisbane. While this template was being written, Milestone 2 files appeared in `entry_plugins/nitro_crt_v2/` (`chop.py`, `backtest_m2.py`, `test_m2_*.py`, plus a changed `plugin.py` and `params_default.json`). They come from separate work and were **not** reviewed here. Refresh this example and the C# port when M2 is logged in VERSION_LOG.

Identity

Field	Value
Trader name	Nitro Trades
Setup / `setup_id`	CRT break: `NITRO_CRT_V2_BREAK` (retest `NITRO_CRT_V2_RETEST` planned for M2)
`source_lane`	`nitro_crt`
Plugin folder (Python)	`smt-python/entry_plugins/nitro_crt_v2/` (v1 `nitro_crt_v1/` archive-only)
NT8 strategy / plugin	`SMT_AR_NitroCRT_v2` / `NitroCrtV2Plugin` (this template; not compiled)
Version / milestone	`crt_v2_m1`: **Milestone 1 done, incomplete** (2026-10-01)
Instruments	MNQ (M1 smoke on NQ prices); MES and MGC planned (F9)
Status	M1. 29 tests pass (box + PC2); smoke PF 0.776 / 0.881, **not a verdict**

Source video

#	Title	Video id	Uploaded	Analysis folder
1	“The only CRT video you’ll ever need…”	`nRIqZwma9xM`	2026-09-28	`/workspace/nitro-setups/nitro_last15_2026-10-01/nRIqZwma9xM/` (`DEEP_CARD.md`, `outcome.json`, `transcript_timed.txt`)

Setup rules

#	Rule	Evidence	Feature
R1	Bias = 1H pivot swept, then a 1H close back (not EMA); optional 1H OB	§2 03:54–04:44, 15:52	F3 (M2)
R2	CRT on clock 15m/30m/1H (ET): C2 sweeps the C1 high (or low) by ≥ 1 tick and closes strictly inside C1; outside bars rejected	§1 overlay	F1
R3	Play the **first** CRT to form (formed_at = C2 close, 10:00–12:00 ET); a tie goes to the lowest TF	09:19	F2
R4	Entry = first 1m break of the C2 far side by 1 tick, inside C3 (expiry 1 TF bar, before 12:00 ET); cancel if the sweep extreme breaks first	§4 08:30, 16:56, 22:17	F4
R5	Target = the opposite C1 extreme; stop built back from TP at **1.5 R:R** (rounded away from entry)	§5 08:42, 17:46, 21:26	F6
R6	One CRT per day (setup eligibility, counted at first	§3, §7	F7

#	Rule	Evidence	Feature
	formation)		
R7	Skip FOMC days and no-news Mondays	§9	F8 (M2)
R8	Watch MNQ+MES+MGC, one trade total	§10	F9 (M3)
R9	Chop on 1m downgrades A+ → A	§3 16:44	F10 (M2)
R10	BE only near target (OM hint)	§5 10:32–11:10	F11 (M3, gap G2)
Grades	A+ = bias + first CRT + clean; A = bias + chop; B = no bias (M1 emits **B**)		

Params with defaults (selection; full list in BUILD_PLAN §3 / `config/params_default.json`)

Param	Default	Feature	Built in M1?
`f1_crt_timeframes_min`	[15, 30, 60]	F1	✅
`f1_htf_anchor` / `f1_htf_min_fill_frac`	clock / 0.5	F1	✅
`crt_min_sweep_ticks` / `crt_close_inside_strict` / `crt_reject_outside_bar`	1 / true / true	F1–2	✅
`crt_min_target_ticks`	8 (**untuned placeholder**)	F1/2/4	✅
`f2_tie_break` / form window	lowest_tf / 10:00–12:00 ET	F2	✅
`f3_bias_*`	pivot_sweep, 2/2, 48h, no-bias → B, counter → skip	F3	❌ (off in m1_smoke)
`f4_entry_mode` / `f4_break_ticks` / `f4_entry_expiry_tf_bars` / `f4_entry_window_end`	break / 1 / 1 / 12:00	F4	✅ (retest ❌)
`f4_cancel_if_sweep_extreme_broken` / `f4_ambiguous_bar_policy`	true / skip	F4	✅
`f6_stop_mode` / `f6_stop_rr`	target_rr / 1.5	F6	✅
`f7_one_crt_per_day` / `f7_count_mode`	true / first_formed	F7	✅
`f8_*` (FOMC skip, no-news Monday)	on	F8	❌ (FOMC file built; CPI/NFP not sourced)
`f9_symbols` / `f9_one_trade_total` / `f9_pick_rule`	[MNQ, MES, MGC] / true / first_trigger	F9	❌
`f10_chop_*`	20 bars, flip 0.6, wick/body 1.5, downgrade	F10	❌
`f11_be_hint_*`	on, 4 ticks	F11	❌
`f12_symbol` / `f12_snap_prices_to_tick`	MNQ / true	F12	✅

Feature → milestone map

Milestone	Features	Status
M1	F1 clock HTF, F2 first-formed, F4 1m break, F6 stop-from-target, F7 one CRT/day, F12 specs	✅ done 2026-10-01
M2	F3 bias, F5 retest, F8 calendar, F10 chop; re-smoke NQ/ES/GC separately	next

Milestone	Features	Status
M3	F9 router (one shared host, per-symbol bars; G1), F11 BE hint (G2), grade ladder	planned
M4	Walk-forward 2/3 IS – 1/3 OOS, both ambiguity policies, sub-minute check, costs, school gate	planned
M5	NinjaTrader port (this template is the scaffold)	deferred

Essentials mapping (`params.essentials_mapping`) and NT8 defaults (`SMT_AR_NitroCRT_v2.SetPluginDefaults`)

Intent	Param	Value	Python	NT8 template
NY AM entries	`AllowOpenWindow`, 09:30 → `f4_entry_window_end`	true, 09:30–12:00 ET	✅	✅
One trade total	`UseOneTradeMode` / `OneTradeMax`	true / 1	✅	✅
No overlap	`Advanced_MaxOpenTrades`	1	✅	✅
Stop / target	`stop_hint` / `target_hint` [C1 opposite]	plugin	✅	✅
Set-and-forget, 1 lot	`ExitMode` / `FixedQuantity`	OneShot / 1	✅	✅
BE near TP	`Advanced_UseBreakEven`	**false** (G2)	partial	off
Trailing	`UseStagedTrail`	false	✅	✅
News guard (OM side)	`Advanced_UseNewsGuard`	unused (G5)	❌	❌
EOD	harness 16:00 ET	`UseEodFlatten` 16:00	harness	✅

Data needs and status (BUILD_PLAN §5, checked 2026-10-01)

Need	Status
NQ 1m (as MNQ)	✅ box `nq_front_month_m1.csv` 2025-07-01 → 2026-09-14 05:58 ET; UTC bar-end → ET start; roll days dropped; 1-dp prices snapped
ES 1m (as MES)	✅ 2025-09-10 → 2026-09-11; `ES_EXCLUDE_DATES`
GC 1m (as MGC)	✅ 2025-12-31 → 2026-09-11; `GC_EXCLUDE_DATES`
Farm PCs	PC2 has stocks only (NQ copied for the run); PC3 offline / unchecked
True micro series	❌
Video replay week 2026-09-14..18	❌ missing for all three
Sub-minute futures	❌ (61 of 131 M1 stops were on the entry minute)
FOMC calendar	✅ `config/calendar/fomc_decision_days.csv` (2025–2027)
CPI / NFP / red-folder	❌ not sourced
CME holidays	❌

Run record

Run	Machine	Data	Policy	Cost	Trades	WR	PF	Net $ (1 MNQ)	Max DD $	Gate
M1 smoke	PC2	NQ→MNQ 2025-	conservative	$0	187	29.9 %	0.776	−2,017.00	2,988.50	sample OK (329

Run	Machine	Data	Policy	Cost	Trades	WR	PF	Net $ (1 MNQ)	Max DD $	Gate
	DESKTOP-09E03F9, 2026-10-01 14:14 AEST	07-01..2026-09-09								days, 187 trades); **incomplete milestone**
M1 smoke	same	same	next-bar	$0	187	41.7 %	0.881	−1,018.00	2,407.50	same
M1 smoke	same	same	conservative	$1.50 RT (assumed)	187	—	0.75	−2,297.50	—	same

Appendix A. Package layout and NT8 import

```
SMT_AllinOne_Roman_NT8_Chassis_Template_v0.1.zip
├── CHASSIS_SPEC_TEMPLATE_v0.1.md / .docx / .pdf
├── trader_plugin_template.yaml
└── NT8/
    ├── README_IMPORT.md
    └── bin/Custom/Strategies/SMT_AllinOne_Roman/
        ├── SMT_AllinOne_Roman_Contracts.cs   (EntrySignal, IEntryPlugin, ClockAggregator, InstrumentSpecs)
        ├── SMT_AllinOne_Roman_Chassis.cs     (abstract base: feed, alignment, OM, risk, params)
        ├── SMT_AR_NitroCRT_v2.cs             (concrete strategy: Nitro CRT v2 M1)
        ├── SMT_AR_BlankTrader.cs             (concrete strategy: blank)
        └── Plugins/
            ├── NitroCrtV2Plugin.cs           (port of crt_v2_m1 detection; stub, untested)
            └── BlankTraderPlugin.cs          (blank stub)
```

Copy `NT8/bin/Custom/Strategies/SMT_AllinOne_Roman/` into `Documents\NinjaTrader 8\bin\Custom\Strategies\` on a **Dev PC**, then compile with F5 in the NinjaScript Editor. Details are in `NT8/README_IMPORT.md`. The files are plain source and are **not** an NT “Export NinjaScript” archive.

Appendix B. Change log

Version	Date (Brisbane)	Change
v0.1	2026-10-01	First template: spec + YAML + NT8 skeleton (not compiled in NT8; box stub syntax check only)