Cascale

HIGG ASSESSMENT MODEL

Higg Assessment Model

Version ham-v0.3.0 · 11 September 2026

The Higg Assessment Model (HAM) is the compiled, evaluator-ready form of the Higg Index calculation logic. A HAM release is a single, versioned bundle: a JSON schema describing every record type, a compressed CBOR payload of the compiled block tree, and a…

Contents

The Higg Assessment Model (HAM) is the compiled, evaluator-ready form of the Higg Index calculation logic. A HAM release is a single, versioned bundle: a JSON schema describing every record type, a compressed CBOR payload of the compiled block tree, and a parallel JSON payload for tools that don't want to deserialize CBOR. Two companion paths consume the bundle: a runtime-embedded path (`usage.md`) where a host process evaluates blocks directly, and an HTTP API path (`guidance-api.md`) where the `stl-local-server` does that work and returns a finished assessment as JSON.

Purpose

HAM exists so the calculation logic for the Higg Index assessments can be:

- **Authored once, in one place.** Cascale maintains the model's source in one place; the release bundle is the compiled form. Consumers don't re-implement the calculations.
- **Versioned independently of platform code.** A new HAM release is a new published bundle; no platform deploy is required to pick up a new model. The server resolves a requested version at request time from a published manifest.
- **Consistent across every consumer.** A facility's score for a given assessment + HAM version is identical regardless of which surface renders it — host platforms, analytics portals, and ad-hoc notebooks pointed at the bundle directly should all produce the same result. Differences in downstream visualisations should never reflect differences in the model.

Versioning

A HAM version is `<major>.<minor>` (e.g. `1.2`). Versions are published into a manifest in the artifact bucket; the manifest is the single source of truth for which versions exist.

Compatibility:

- **Patch-level changes** overwrite the existing `<major>.<minor>/` prefix in place. Running servers pick them up on the next refresh tick (default 60s). Reserved for purely additive changes — content fixes, new emission factors, additional record fields that older consumers can safely ignore.
- **Minor releases** publish a new `<major>.<minor>/` prefix and extend the manifest. New record types or new required schema properties land here. Consumers should treat the version integer pair as opaque — pin to a specific version, fetch the schema, and use whatever's there.
- **Major releases** are reserved for breaking schema changes (a record removed, a field's `type` narrowed, an `evaluatedType` semantic change). Consumers are notified before a major lands; a migration window is announced in the release notes.

Channels: in addition to numeric versions, a release may pin a named channel — most commonly `next`, which always points at the latest in-progress release. `next` is appropriate for staging / preview environments; production should always pin a numeric version.

A request that asks for `1` (the major alone) resolves to the highest numeric minor under `1.*`. A request that asks for `next` resolves only to the named channel — never to a numeric version. This means `1` will not surface a `next`-channel release until that release becomes `1.<n>`.

Content

A HAM bundle contains the following files. The bundle is laid out flat in the release artifact directory and the same set of files is distributed under each `ham/<version>/` prefix in the artifact bucket.

- **`READ_ME_FIRST.md`:** Routing index. The right place to start when opening a fresh release.
- **`model.cbor.gz`:** The primary artifact. CBOR-serialized `Artifact` struct, gzip-compressed. Consumed by `stl-local-server` and any other Rust/typed-deserialization path.
- **`model.json.gz`:** The same logical content as `model.cbor.gz`, expressed as JSON. Intended for `jq`, Python notebooks, JS embedded evaluators, and anything else that prefers JSON over CBOR. Slightly larger on disk.
- **`schema.json`:** The bundle's JSON schema as raw JSON. Use this to validate that a candidate input conforms to the model, or to drive code-generation in a typed consumer.
- **`schema.md`:** A human-readable, auto-generated rendering of the schema — record types, field descriptions, expected types after JS expression evaluation. Read this when you want to understand the shape of the data without machine-parsing the JSON schema.
- **`usage.md`:** Implementer guide for the runtime-embedded path — how to consume `model.json.gz` directly in your own evaluator. Use this when your application wants to run HAM in-process.
- **`guidance-api.md`:** Integrator guide for the HTTP API path — how to call `stl-local-server` to evaluate assessments. Use this when your application wants to delegate evaluation to a HAM-versioned API.
- **`guidance-ham.md`:** This file.
- **`guidance-ghgp-fem.md`:** Methodology guide for the GHGP-compliant energy and emissions calculations the model produces for FEM cadences.
- **`guidance-eeci.md`:** Methodology guide for the Effective Energy Carbon Intensity metric.
- **`guidance-fep.md`:** Methodology guide for the Foundational Environmental Performance overlay.

Glossary

Terms that appear across the bundle's docs.

- **HAM** — Higg Assessment Model. This document set + the released bundle of compiled logic.
- **FEM** — Higg Facility Environmental Module. The annual environmental assessment. Each year's edition is a *cadence*: `fem2024`, `fem2025`, etc.
- **FDM** — A monthly FEM-variant assessment, cycling one year behind the matching FEM cadence (`fdm2024` is the monthly follow-on for `fem2023`'s reporting year).
- **BRM** — Higg Brand and Retail Module. A separate annual cadence with its own STL tree and its own release flow.
- **FSLM** — Higg Facility Social and Labor Module. Annual social audit produced by layering Cascale scoring on a SLCP-authored source workbook (see *CAF*). FSLM cadences look like `fslm_1_7`, `fslm_2_0`. **Not modelled in HAM.** FSLM ships its own workbook artifact through a separate release pipeline.
- **CAF** — Common Assessment Framework. The SLCP-authored source workbook that FSLM scores. CAF and FSLM are distinct artifacts: CAF is the input, FSLM is Cascale's scored output.

- **rfi_pid** — The internal identifier for a cadence, e.g. `fem2025` or `fslm_1_7`. Stable across years; downstream consumers pin against rfi_pids, not display names.
- **GHGP** — Greenhouse Gas Protocol. The accounting standard FEM's energy and emissions calculations follow. See `guidance-ghgp-fem.md`.
- **EECI** — Effective Energy Carbon Intensity. A facility-level decarbonization metric Cascale publishes from FEM data. See `guidance-eeci.md`.
- **FEP** — Foundational Environmental Performance. A tag overlay that marks a subset of FEM Level 1 questions as foundational. See `guidance-fep.md`.
- **STL** — Structures Template Language. The internal DSL the calculation logic is authored in. STL is a build-time concern only — the compiled bundle hides it. Mentioned in the bundle docs only when explaining the provenance of a calculation.
- **Block** — A compiled record. Every entry in `model.json.gz` under `blocks.<rfi_pid>[]` is a block. The `__type__` field identifies which schema record type it instantiates.
- **Scope (1 / 2 / 3)** — The GHGP categorisation of emissions. Scope 1 is direct; Scope 2 is purchased energy; Scope 3 is other indirect. FEM models Scope 1 and Scope 2 only.
- **Market-based / location-based** — The two Scope 2 accounting methods. See `guidance-ghgp-fem.md`.

Where to go next

The right next-page depends on what you're trying to do:

- *Run HAM against an assessment via HTTP?* Go to `guidance-api.md`. That's how most integrations consume HAM.
- *Embed HAM directly into a host process?* Go to `usage.md`. The bundle's `model.json.gz` plus an STL-style block evaluator gives you the same result the API would.
- *Interpret the values that come back out?* Go to the methodology guides: `guidance-ghgp-fem.md`, `guidance-eeci.md`, `guidance-fep.md`. These explain what the calculations mean, what assumptions they encode, and what you should and shouldn't do with the output.
- *Understand the shape of a particular record type?* Go to `schema.md`, which is the human-readable schema dump. The machine-parseable form is `schema.json`.