Shorthand Note: Overarching Formalism & Operationalization of the Multisensor Fusion Lake

1 Overarching Formalism

Metadata–Signal Factorisation. Every geospatial measurement is treated as a pair

$$(\underbrace{m}_{\text{metadata}}, \underbrace{s}_{\text{signal}}) \in \mathsf{Meta} \times \mathsf{Signal},$$

where Meta is a STAC[1] object and Signal is a Zarr store. The product structure guarantees that changes in calibration, reprojection grids, or statistical models never mutate the physical bytes of other factors.

Sensor Alignment. Let $S_i : \mathcal{G} \to \mathbb{R}^{k_i}$ be the field generated by the i-th sensor after applying its affine reprojection T_i^{-1}. With m sensors the aligned bundle is

$$\boldsymbol{S}(\boldsymbol{x}) := \big(S_1(\boldsymbol{x}), \dots, S_m(\boldsymbol{x})\big), \qquad \boldsymbol{x} \in \mathcal{G} \subset \mathbb{R}^3.$$

Fusion Functional. For a task–specific function $g : \mathbb{R}^{k_1+\cdots+k_m} \to \mathbb{R}^d$ we estimate the posterior mean

$$F(\boldsymbol{x}) \;=\; \mathbb{E}\big[g\big(\boldsymbol{S}(\boldsymbol{x})\big) \;\big|\; \text{data, hyper–priors}\big],$$

evaluated chunk–wise so that computation factorises over Zarr keys.

Reproducibility Ledger. Each execution appends

$$\Gamma := \big[t,\ \text{item\_hash},\ \text{posterior\_hash},\ \text{ELBO},\ E,\ \tau\big]$$

to an append–only log; cryptographic hashes make the entire pipeline provably referentially transparent.

2 Operationalization

2.1 Physical Layout (S3 / GCS / MinIO)

```
s3://fusion-lake/
  raw/            # original sensor blobs
  stac/           # catalog, collections, items
  posterior/      # derived Zarr groups (versioned)
```

[1] SpatioTemporal Asset Catalog.

2.2 Ingestion DAG

1. **Stage**: wrap new object in a STAC `Item`; attach checksums.
2. **Reproject**: convert native grid $\rightarrow$ common $\mathcal{G}$; write `asset.type = application/vnd+zarr`.
3. **Posterior Notebook**: parameterised execution (Papermill) computes (μ, σ^2) or other sufficient statistics; emits new Zarr under `posterior/`.
4. **Ledger Write**: hash `Item` and Zarr root; append to Γ.

2.3 STAC $\leftrightarrow$ Zarr Contract

- Every STAC asset whose `type` is `application/vnd+zarr` *must* resolve to a store whose root contains the JSON key `"zarr_format"`.
- A STAC extension field `"xarray:open_kwargs"` specifies the loader's arguments, enabling pure–metadata discovery by Jupyter / LangChain agents.

2.4 Notebook Template (Skeleton)

```
cat = intake.open_stac_catalog("s3://fusion-lake/stac/catalog.json")
item = cat["sentinel-2"][ITEM_ID]
ds   = item["B04_zarr"].to_dask()   # lazy xarray
# --- science here ----------------------------------------------------
posterior = bayes_update(ds)
posterior.to_zarr("s3://fusion-lake/posterior/ndvi_v0.zarr", mode="w")
```

2.5 Key Guarantees

Immutability All STAC Items are write–once; updates create new Item IDs.

Chunk Locality Fixed 512×512 window $\Rightarrow O(1)$ object–store reads.

Schema Versioning Evolution of Meta or Signal is tracked by semantic version tags in `stac/collections`.