4D-STEM Requirements Specification

Ciaran Welsh / SEM Team

August 14, 2025

Contents

1 Introduction

This document defines the implementation-independent software requirements for 4D-STEM users. It is a reference for software and application engineers as well as the business side of ASI. The document should be considered 'living' in that the information should be updated with pertinent information as the project evolves.

4D-STEM (Four Dimensional Scanning Transmission Electron Microscopy) is a microscopy technique where a focused electron beam is scanned across a sample, producing a 2D diffraction pattern at each position. The resulting four-dimensional dataset, with two spatial and two reciprocal-space dimensions, enables detailed analysis of structure, strain, and other material properties. The advantage that Timepix3 (and Timepix4) offer such an application are derived from the event-driven nature of the detector; timepix detectors are event driven and therefore record each hit independently from other pixels, and therefore avoiding some of the frame-based limitations and offering greater performance. With tpx3 in 4D-STEM we can get small dwell times, improved dose efficiency for beam-sensitive samples, reduced data volume and supports real-time analysis with lower memory and bandwidth requirements.

From a business perspective, the primary project goal is to sell more ASI detectors by embedding them into the 4D-stem solution. The 4D-STEM component adds considerable value to the detector and therefore makes this a worthy enterprise for ASI. Our goal is to build a full 4D-STEM solution, including both hardware and software components. Importantly we design for and include the next generation of timepix (Timepix4) into the design from day 1. We might not be able to support the full rate timepix 4 from the begining but with appropriate modularity in the design we should be able to drop in higher performance replacements for performance critical components (such as versions that run on the GPU).

To date, ASI have focused their efforts on building this feature into Accos and Serval, but with difficulties. This document is not intended to be a review of that work, but suffice to say that the existing solution is not an ergonimic experience for neither the user nor the developer. The proposal outlined in this document is intended as a replacement to the 4D-STEM corners of Accos and Serval.

In the following sections we describe the system we intend to build from a high level perspective. We describe the envionment the system needs to run, including software and hardware dependencies and we outline the needs of the end user that this software aims to solve. We outline functional requirements and the architectural characteristics that developers need to design for, based on these needs.

2 Overall Description

The primary function of this software is to enable users one integrated workstation for the control of their detector and scan generators. The software needs to be one cohesive unit that enables scientists to conduct their experiments and collect data for further analysis. It is the first tool in their chain of tools that delivers them results for publications. As such this software needs to deliver ergonimic workflows for the evaluation and acquisition of their sample. It needs a performant live image stream to provide microcopists with enough information to evaluate a section of sample and interoperability with other tools that enable further study of their data, including LiberTEM, HyperSpy and py4dStem. We do not focus on specialised downstream analysis techniques such as ptychography.

The software must be suitable for execution on a single workstation. Serval is the primary mechanism for controlling the detector and we need multiple mechanisms of control for a choice of scan generators to reach a larger audience. From the user perspective however, the new software is the one over-riding central point of control for all the tools necessary to conduct their experiment. A general overview of the application's position in the broader ecosystem is shown in fig. 1.

External connections are:

- **Serval**: REST API for detector configuration and TCP stream for event data.
- **Scan generator**: Controlled via vendor-supplied shared library (FFI), gRPC or another system of communication.
- **Local storage**: High-throughput NVMe drives for data capture.

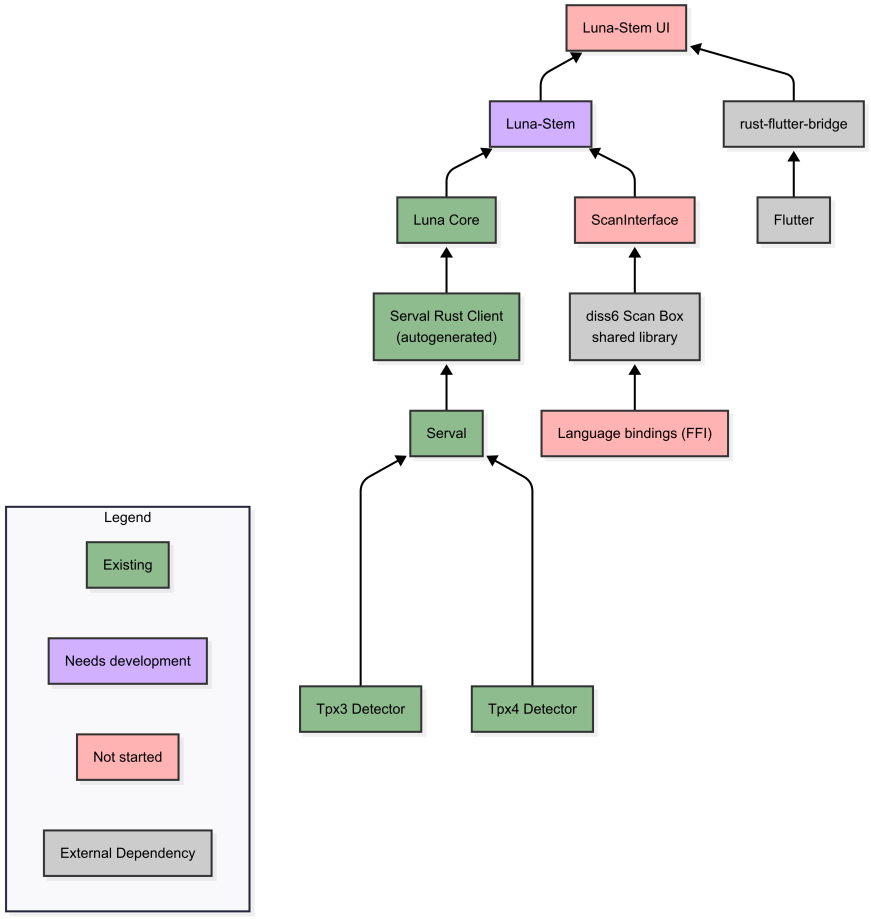

Figure 1: Overview of the application in relation to other tools

2.1 User Classes and Characteristics

The primary users are 4D-STEM scientists, who also operate the microscope. Users range from computationally literate power users to those without much experience. Both users are mainly focused on scientific outcome rather than on the processing required to get them there. The software abstracts data acquisition and visualisation details so that no programming skills are required for them to do just that.

2.2 Definitions and Acronyms

1. ¡expand and define¿
2. CBED:
3. vADF:
4. iDPC:
5. iCOM:
6. Diffraction bands: the amount of diffraction an electron experiences is related to the charge of the atoms inherent in the sample (please validate this statement). A typical sample therefore has characteristic diffraction patterns which can be selected using an annulus shape.

2.3 Operating Environment

The application supports Windows, Ubuntu, and macOS at launch; other Linux distributions can be built on request. Our primary system is ubuntu and several aspects of our tooling are already optimized for ubuntu. We therefore aim for performance targets primarily on ubuntu, and only secondarily on the other operating systems. The software shall run on a high-performance lab workstations with specifications sufficient for sustained high-rate data capture and processing. GPU acceleration is planned for future versions but is not required initially. The system operates on localhost only, with all components running on the same machine. Storage must sustain the highest practical write speeds for the selected output format.

2.4 User Needs

1. **Seamless integration with Serval for detector control whilst being exposed to only what they need**
   Rationale: Serval must be used for detector control but minimizing the user interaction with Serval reduces the learning curve and enhances usability.

2. **Real-time visual feedback during scans**
   Rationale: Enables rapid navigation to regions of interest and immediate quality assessment. Also prevents wasted beam time.

3. **Ability to run essential offline analyses inside the application, such as offline image reconstruction methods**
   Rationale: Enables quality assessment decisions without exporting large datasets to external tools.

4. **Acquire continuous or multi-frame ('Z-stack') scans**
   Rationale: Lets users monitor radiation damage frame-by-frame and perform statistics based on multiple measurements in a region.

5. **Flexible detector masking (hardware or software) to select pixels for subsequent analysis.**
   Rationale: Enables selection of diffraction bands for online and offline image reconstruction algorithms.

6. **Switchable scan modes to balance performance and user feedback trade-offs**
   Rationale: Allocation of computational resources should be allocated differently depending on whether the user is searching for interesting features or whether they have already found them and want to acquire data for further analysis.

7. **Export raw *or* processed data in open formats (e.g. SparseArray, HDF5, Zarr) and where appropriate, image data**
   Rationale: Guarantees images can be saved and interoperability with downstream analysis suites such as LiberTEM, HyperSpy, and py4DSTEM.

8. **Real-time observability of acquisition-pipeline health (hit-rate, processing load, synchronization)**
   Rationale: Gives user vital feedback that can be used to understand what is happening in the pipeline. This includes amounts of dropped data, per chip level hit rates and data throughput/latency rates.

9. **Dedicated 4D-STEM interface separate from the generic detector UI**
   Rationale: Build with a 4D-stem first mentality. Makes software 4d-stem centric, minimises cognitive load on our users needing to compete with more general use cases of the timepix3 detector and prevents accidental changes to routine detector operations. Separation of concerns.

10. **Immediate and clearly organized access to scan-generator controls**
    Rationale: Scan controls are some of the most important in 4D-STEM, they should be prevalent and obvious in the user workflow for better user experience, operating on the principle of least clicks to control your scan.

3 System Features and Requirements

3.1 Functional Requirements

3.1.1 Serval Interaction Layer

Serval is the control interface between the application and the detector.

1. Must **REQ-SRV-01 – Attach or spawn Serval.** On startup, the system shall either (a) attach to a reachable Serval instance at the configured host:port, or (b) spawn Serval as a child process whose lifetime is tied to the 4D-STEM UI.

   Acceptance: Attach succeeds or a child process is started; ownership is recorded in metadata. If attached, the UI does not terminate Serval on exit; if spawned, the UI terminates it cleanly on exit.

   Verification: Start with/without reachable Serval. Check attach vs spawn, and exit behavior. Fixture: A mini server that looks like Serval for the parts that matter (dashboard + status).

2. Must **REQ-SRV-02 – Version negotiation.** On initial connect, the system shall retrieve Serval's API version and compare it to the supported range. If out of range, the UI shall warn; if marked incompatible, operations requiring those features Could be blocked.

   Acceptance: Version captured in logs/metadata; Could correct warning/blocking per compatibility matrix.

   Verification: Test: emulate 'too old/too new/incompatible' versions; assert warning banner and Could blocked actions.

3. Must **REQ-SRV-03 – State subscription (push).** The client shall subscribe to Serval state changes (e.g., detector {Disconnected, Connected, Idle, Running}); Won't periodic polling for state shall not be used.

   Acceptance: State transitions correctly arrive

   Verification: Test: push state changes from stub/mock serval; capture timing

4. Must **REQ-SRV-04 – Detector-connected init sequence.** Upon receiving *detector_connected*, the system shall perform: (1) upload DACs, (2) upload bad-pixel map (BPC), (3) set data destination, (4) load saved detector settings (e.g. bias) for this detector chipboard Id. Failures stop the sequence and surface a recoverable error.

   Acceptance: All four steps succeed, or the sequence aborts with a clear message; successful steps are idempotent (has no effect) on retry.

   Verification: Test: inject failure at each step; assert stop + message; rerun to confirm idempotency of prior steps.

5. Must **REQ-SRV-05 – Command error model.** All Serval command failures shall be represented or mapped to a structured error with code/type, human message, machine key, and remediation hint.

   Acceptance: Errors in logs and UI contain structured fields; API errors use the documented schema.

   Verification: Inspection/Test: trigger known errors; assert schema fields present and logged.

6. Must **REQ-SRV-06 – Connectivity loss during acquisition.** If the control connection to Serval drops during an active acquisition, the client shall detect loss, attempt reconnect, re-establish subscriptions, and abort the acquisition gracefully.

   Acceptance: Loss detected; after recovery, state is consistent the error is recorded in logs and the user is notified

   Verification: Manual test: remove the ethernet cable mid run; verify abort gracefully path. Validate logs.

7. Must **REQ-SRV-07 – Logging and correlation.** Every Serval command, response, state change, and error shall be logged in the serval http log stored per workspace

   Acceptance: Logs contain request/response pairs

   Verification: Send a command; locate the log and validate

8. Must **REQ-SRV-08 – Transport keepalives.** The client shall use protocol keepalives to detect dead connections without polling for state.

   Acceptance: Dead connections are detected within the configured interval; reconnect attempts follow.

   Verification: Test: drop TCP silently; observe keepalive detection timeline and retry behavior.

9. Must **REQ-SRV-09 – Endpoint configuration.** Serval host, port, and transport shall be configurable in the UI; changes take effect on "accept" button press.

   Acceptance: Editing config changes the target; connections establish to the new endpoint after reconnect.

   Verification: Test: change the end point to another socket running a different serval instance. Verify connectivity.

10. Won't **REQ-SRV-10 – Equalization in client.** The client shall not perform detector equalization; that function remains in Accos.

    Acceptance: No equalization commands exist; UI offers no equalization controls.

11. Won't **REQ-SRV-11 – Detector orientation in client.** The client shall not manage detector orientation.

    Acceptance: No orientation commands exist; UI offers no orientation controls.

    Verification: Inspection.

3.1.2 Online Image Reconstruction

1. Must **REQ-OIR-01 – Real-time vADF reconstruction.** Raw event data received from Serval via TCP shall be processed in real time into images using the vADF algorithm. Processing is performed by *luna-stem/luna-core* and displayed in the 4D-STEM UI. For selection of pixels that contribute to final vADF image, an annular ROI shall be used (item 16)

   Acceptance: .

   For dwell times $\geq 1\ \mu s$ at 320MHz per second the image shall display cleanly and without lagging or losing data. *It is unknown whether this target is realistic* [Test: replay a recorded 512×512, $1\,\mu s$ dwell event stream and measure performance]

2. Must **REQ-OIR-02 – Preview non-interference.** The preview function has a cost so it inevitably reduces other taks that the computer can do at the same time. However the preview function must not reduce performance to such a point that data integrity is compromised.

   Acceptance: With preview enabled, acquisition hit-rate, mapping accuracy, and dropped-event counts remain within $\pm x\%$ (e.g. some currently unknown percentage) of preview-disabled baseline on identical runs.

   Verification: Test: Compare runs between the preview on/off with identical replay date; compare hit-rate, mis-maps, and drop data; assert within tolerance.

3. Should **REQ-OIR-03 – GPU-accelerated vADF.** Provide a GPU-accelerated implementation of vADF on supported hardware. Especially when targeting tpx4 (in future) which has x5 the performance requirements of a tpx3 turbo, a GPU backend should be built

   Acceptance: On compatible GPUs, performance increase by $\geq x\%$ vs. CPU-only under the same conditions; outputs match CPU within tolerance. A realistic and precise value of x is currently unknown.

   Verification: Test: run CPU vs. GPU on same dataset; measure latency reduction; ensure the same results are achieved.

3.1.3 Offline Image Reconstruction

1. Must **REQ-OFR-01 – Single CBED display.** When a user selects a probe position, the system shall display the corresponding CBED diffraction pattern from disk.

   Acceptance: Image is computed from offline data and displayed within a reasonable time (100ms max)

   Verification: Test: simulate a simple known dataset and build a pixel for pixel image (256x256) of the detector using 1 for hit and 0 for not hit. Compare pixels that have intensities against those the test dataset.

2. Must **REQ-OFR-02 – Average CBED (all).** The system shall compute and display the Average CBED over all probe positions in the scan.

   Acceptance: Image is computed from offline data and displayed within a reasonable time (100ms max)

   Verification: Test: simulate a simple known dataset and build a pixel for pixel image (256x256) of the detector using 1 for hit and 0 for not hit. Compare pixels that have intensities against those the test dataset.

3. Must **REQ-OFR-03 – Offline vADF.** The system shall recompute vADF from stored data by summing intensities within the selected detector region for each probe position. The selection ROI shall be annular.

   Acceptance: Image is computed from offline data and displayed within a reasonable time (100ms max)

   Verification: Test: simulate a simple known dataset and build a pixel for pixel image (256x256) of the detector using 1 for hit and 0 for not hit. Compare pixels that have intensities against those the test dataset.

4. Must **REQ-OFR-04 – iCOM.** Provide offline iCOM reconstruction on supported datasets. Performance must be ”reasonable” but not heavy duty, like the online algorithms

   Acceptance: Image is computed from offline data and displayed within a reasonable time (100ms max)

   Verification: Test: run iCOM; verify axis metadata, units; compare against a validated reference implementation on a fixture.

5. Should **REQ-OFR-05 – iDPC.** Provide offline iDPC reconstruction on supported datasets. Performance must be ”reasonable” but not heavy duty, like the online algorithms

   Acceptance: Image is computed from offline data and displayed within a reasonable time (100ms max)

   Verification: Test: run iDPC; verify axis metadata, units; compare against a validated reference implementation on a fixture.

6. Must **REQ-OFR-06 – Asynchronous execution + progress.** Offline operations shall run asynchronously with progress reporting and possibility of user cancellation. Partial outputs are marked incomplete with a warning and message.

   Acceptance: Long-running tasks (>500 ms) show progress; cancel stops computation leaves no corrupt outputs. Failure handled gracefully.

   Verification: Test: start each algorithm; observe progress; cancel mid-run; assert timely stop and clean state.

7. Could **REQ-OFR-07 – Random-access readiness.** If needed for performance, maintain an index that maps probe position to data blocks/offsets next to processed data output to facilitate fast lookup and retrieval of data for building images (such as CBED).

   Acceptance: Index builds after acquisition in the background and stored on disk for reuse.

   Verification: Test: Index identifies correct hits and indexing occurs in ’reasonable‘ time.

8. Won't **REQ-OFR-08 – Ptychography (offline).** Offline ptychography is out of scope.

   Acceptance: No UI/CLI entry points for ptychography exist.

9. Won't **REQ-OFR-09 – 4D-STEM EELS (offline).** Offline 4D-STEM EELS is out of scope.

   Acceptance: No UI/CLI entry points for EELS exist.

3.1.4 Z-stacking Frames

1. Should **REQ-ZST-01 – Fixed-length Z stack.** When the user sets $Z > 1$ and presses *Start (Acquire)*, the system shall execute exactly Z full probe-grid scans. Every detector event shall be assigned to frame and probe position.

   Acceptance: Exactly Z completed frames are recorded; for each completed frame, all expected probe coordinates are present once.

   Verification: Test: run $Z = 3$ on a test dataset; verify frame count, per-frame coordinate coverage, and sorter counters; assert zero mis-assign/unassigned.

2. Should **REQ-ZST-02 – Continuous Z mode.** When Continuous Z mode is selected, the system shall repeat full probe-grid scans until the user issues *Stop* or an automatic stop condition occurs (e.g., disk guard). Frames in continuous mode shall use a monotonically increasing *frame_id* $\in \{0, 1, 2, \dots\}$.

   Acceptance: Acquisition runs until Stop or disk guard triggers; no partial frames are reported as complete; the final manifest lists all completed *frame_id* values consecutively without gaps.

   Verification: Test: run 60 s continuous; issue Stop; validations

3. Should **REQ-ZST-03 – Frame output writing.** Frame data shall be written to disk in the user-selected format(s).

   Acceptance: After power-loss or forced abort during a frame, partial frame data is "salvaged".

   Verification: Test: fault-inject mid-frame (kill process) and verify.

3.1.5 Output and Raw Data Collection

1. Could **REQ-OUT-01 – Acquire preview policy.** If necessary, and depending on the performance characteristics of the system (yet to be determined), the final acquisitions shall run with preview disabled by default. If explicitly enabled, the user shall be warned before they acquire that preview + acquisition on the same computer can lead to data loss.

   Acceptance: Run (Acquire) with and without the preview pipeline. Measure performance metrics.

2. Must **REQ-OUT-02 – Raw data immutability.** Raw event data shall be written to disk exactly as received from Serval (ordering and content preserved), with no corrections applied in-line by the client or Serval.

   Acceptance: Raw-file checksums match a capture of the a known dataset

3. Must **REQ-OUT-03 – Corrections policy.** Corrections required for scientific correctness (e.g., tram-lane and PLL corrections) shall be applied only in reconstructions (online/offline), never to raw. Correction parameters are retrieved from Serval (or calibration files) and recorded in provenance.

   Acceptance: Raw remains unmodified; derived outputs include correction names, versions, and parameters; results match a reference .

   Verification: Test: run offline vADF/iCOM with/without corrections; verify provenance; compare to reference.

4. Won't **REQ-OUT-04 – Timewalk correction in raw.** Timewalk correction shall not be applied to raw event data. The time resolution is usually not small enough to have a huge impact.

   Acceptance: No timewalk steps present in raw pipeline; any timewalk handling appears only in derived workflows (if ever added).

   Verification: Inspection.

5. Must **REQ-OUT-05 – Sparse event export (LiberTEM).** Provide a sparse event export compatible with LiberTEM (indices/values plus shape/coords).

   Acceptance: Exported dataset opens in LiberTEM and yields expected hit counts/statistics.

   Verification: Test: manual load in LiberTEM; compare counters to manifest.

6. Should **REQ-OUT-06 – Zarr export.** Provide a Zarr export for 4D data (e.g., $[H, W, Q_x, Q_y]$) with chunking and compression per DATA-FMT; write online if feasible, else offline converter.

   Acceptance: Zarr opens in supported Python stacks (dask/xarray); shape/chunks/attrs match spec.

   Verification: Test: open with xarray; check attrs; compare slices to reference.

7. Should **REQ-OUT-07 – HyperSpy HDF5 export.** Provide HyperSpy-compatible HDF5 export for reconstructed data; write online if feasible, else via offline converter.

   Acceptance: File loads in HyperSpy; axis metadata/units correct.

   Verification: Test: open in HyperSpy; verify axes/units; compare numerics.

8. Must **REQ-OUT-08 – Image export (2D).** Any 2D image produced (e.g., vADF, CBED, iCOM maps) shall be exportable to TIFF and PNG; 16-bit TIFF supported; metadata embedded where format allows.

   Acceptance: Exports succeed; TIFF bit depth correct; embedded tags include units and scaling;

   Verification: Test: export samples; read back; verify tags/bit depth and pixel stats of known dataset. Visually look at the images..

9. Could **REQ-OUT-09 – ROI-based export.** For data reduction, users may export subsets (ROIs in real space or diffraction space) to supported formats without duplicating full datasets.

   Acceptance: Exports contain only selected indices/pixels; metadata record the parent dataset and ROI definition.

   Verification: Test: export ROI; verify indexing and metadata linkage.

10. Could **REQ-OUT-10 – Offline format converters.** Provide CLI/GUI converters among supported formats

    Acceptance: Converters preserve values; progress + cancellation supported.

    Verification: Test: round-trip conversions; compare checksums/tolerances; cancel mid-run.

11. Must **REQ-OUT-11 – Directory layout and naming.** Acquisitions shall use a standardized directory layout and naming convention. Users can modify their pattern with a selection of template strings. Datetime, run ID, sample ID are examples.

    Acceptance: Created structure matches spec; names are unique and sortable.

    Verification: Inspection/Test: create run; validate layout against schema.

12. Could **REQ-OUT-12 – Compression and chunking.** For formats supporting it (Zarr/HDF5), chunking and compression shall be implemented for offline use.

    Acceptance: Resultant compressed data shall exist after

    Verification: Test: round trip must work as expected

3.1.6 Pixel Masking

1. Must **REQ-PMK-01 – Software masking.** The system shall support software inclusion zones or 'mask' defined on the detector pixel grid (rectangular, circular, and annular at minimum). Software masks are applied in image reconstruction; raw data remain unchanged.

   Acceptance: Masks can be created/edited/removed when idle; during acquisition masks are immutable. Mask definitions (shape + parameters + schema version) are stored in measurement metadata and honored by online/offline reconstruction.

   Verification: Test: define each shape; acquire; verify masked regions contribute zero; inspect metadata JSON for shapes/schema.

2. Could **REQ-PMK-02 – Hardware masking.** The system shall support hardware masking by uploading a pixel configuration via Serval.

   Acceptance: After applying a hardware mask, hit counts from masked pixels drop to zero.

   Verification: Test: apply/revert mask via API; compare pre/post hit maps; assert zero hits in masked region.

3. Must **REQ-PMK-03 – Mask mode selection.** If both hardware and software masking are available, the UI shall allow selecting the active mode; switching is permitted only when idle.

   Acceptance: Exactly one mode is active; controls are disabled during acquisition; mode is recorded in metadata.

   Verification: Test: toggle modes while idle vs. running; assert prevention whilst acquiring and metadata.

4. Must **REQ-PMK-04 – Mask visualization.** Masks shall be visibly overlaid on the viewport; hardware and software masks use distinct visual styles.

   Acceptance: Overlay shows correct bounds; legend indicates active mode; visibility toggle works.

   Verification: Inspection/Test: draw masks; verify overlay and legend.

5. Must **REQ-PMK-05 – Autoload masks.** When loading an existing dataset/measurement, previously saved masks shall autoload and appear in the UI.

   Acceptance: Opening a dataset renders masks from metadata; reconstruction honors them.

   Verification: Test: save with masks; reopen; verify overlay and effect.

3.1.7 Acquisition Modes

1. Must **REQ-ACQ-01 – Search mode.** Search mode shall render real-time preview (vADF) and, by default, shall not persist raw events or reconstructions.

   Acceptance: In Search, no raw files are created.

   Verification: Test: run Search; verify absence of files and presence of preview vADF.

2. Must **REQ-ACQ-02 – Acquire mode.** Acquire mode shall persist raw data data to disk in Sparse matrix format. Preview is disabled by default; if enabled, it must not interfere with data acquisition.

   Acceptance: Files are created; preflight disk guard enforced; preview (if enabled) meets works without hurting data integrity.

   Verification: Test: run Acquire with/without preview.

3. Must **REQ-ACQ-03 – Mode transitions.** Mode selection shall be explicit and visible; mode changes are blocked while an acquisition is active.

   Acceptance: Attempting to change mode during an active run is prevented with a clear message.

   Verification: Test: switch modes mid-run; assert block + message.

4. Should **REQ-ACQ-04 – Parameter locking in Acquire.** On *Start (Acquire)*, scan parameters and masks are locked for the duration of the run.

   Acceptance: Controls affecting acquisition parameters become read-only; changes take effect only on the next run.

   Verification: Test: attempt edits during Acquire; assert no effect.

3.1.8 Scan Generator

1. Must **REQ-SCN-01 – Backend selection.** The system shall support selecting a scan-generator backend from a configured list; unavailable backends appear disabled.

   Acceptance: All backends work with their respective scan generators when connected

   Verification: Test: scan integration test suite should pass with any backend.

2. Must **REQ-SCN-02 – Start-time synchronization.** Detector acquisition start and scan start shall be synchronized at the start-of-scan boundary. How do we measure this?

   Acceptance: On qualified hardware, measured start skew $|\Delta t| \leq 1\,\mu s$ over 100 trials; failures surface as warnings and block Acquire if policy requires.

   Verification: Test: instrument start-of-scan and detector-gate signals; compute skew distribution; assert bound.

3. **REQ-SCN-03 – ROI/geometry translation.** Scan generator shall accept ROI-derived geometries and produce matching probe grids for raster scan.

   Acceptance: Generated grids match requested $H \times W$ (per rounding rules).

   Verification: Test: send multiple ROIs; verify grid coverage.

4. Must **REQ-SCN-04 – Graceful stop.** On *Stop*, the scan generator shall honor the 'stop now' and 'stop at frame boundary' semantics from Z-stacking.

   Acceptance: Observed behavior matches selected stop mode.

   Verification: Test: stop mid-frame with both options; verify manifests.

3.1.9 Observability and Pipeline Health

1. Must **REQ-OBS-01 – Metric definitions.** Performance metrics (hit-rate, throughput in bits/s, ingest→present latency) shall be computed per Appendix

   Acceptance: Implementations use the formulas and windows; unit tests validate computations on fixtures.

   Verification: Test: feed synthetic counters; verify computed metrics match reference.

2. Must **REQ-OBS-02 – Aggregation levels.** Statistics shall be computed for the whole detector and per-chip.

   Acceptance: UI/API exposes both aggregation levels; chip IDs match detector layout.

   Verification: Test: compare totals to sum of chips; assert equality.

3. Must **REQ-OBS-03 – Hit-rate UI cadence.** While Running or Previewing, the UI shall update hit-rate at least once per second.

   Acceptance: Observed update interval $\leq 1.0\,\text{s}$ over a 60-s run.

   Verification: Test: capture UI updates; compute intervals.

4. Should **REQ-OBS-04 – Per-stage throughput (opt-in).** When observability is enabled, each pipeline stage shall report throughput in bytes/s and queue depths; this view is off by default.

   Acceptance: Opt-in UI shows per-stage throughput and queues; disabling hides it; enabling does not exceed overhead limits.

   Verification: Test: toggle feature flag; verify metrics appear/disappear; check overhead.

5. Must **REQ-OBS-05 – Latency reporting.** The system shall compute ingest→present latency (and ingest→disk latency for Acquire) and update a 1-s rolling mean every 0.5 s.

   Acceptance: Numbers match PERF-DEF computations; update cadence holds.

   Verification: Test: replay fixture; verify rolling stats and cadence.

6. Must **REQ-OBS-06 – Observability overhead bound.** With observability enabled, median ingest→present latency shall not degrade by more than +10 ms vs. baseline.

   Acceptance: A/B runs differ by ≤ 10 ms median; throughput unchanged within $\pm 0.1\%$.

   Verification: Test: A/B toggle; assert limits.

7. Could **REQ-OBS-07 – Scan–detector sync metric.** Provide a sync metric estimating jitter between scan-box timing and detector hit timestamps, e.g., $J = \text{abs}(t_{\text{hit@40MHz}} - t_{\text{scan}})$, with histogram and summary stats.

   Acceptance: Metric and histogram available when hardware exposes both clocks; results recorded in telemetry.

   Verification: Test: on capable hardware, compute J; verify histogram export.

8. Must **REQ-OBS-08 – Dropped-data visibility.** Dropped/lost/mis-mapped event counters shall be tracked per second and surfaced in UI and logs, with clear warnings when non-zero.

   Acceptance: Injected drops produce visible warnings and non-zero counters; counters reset per run and are written to manifest.

   Verification: Test: inject drops; verify UI/log/manifest.

3.1.10 Data Integrity at Extreme Rates

1. Must **REQ-INT-01 – Mapping accuracy at max qualified rate.** For hit-rate ≤ 320 Mhits/s and dwell $\leq 1\,\mu s$, the system shall assign $\geq 99.999999\%$ of detector hits to the correct probe coordinate.

   Acceptance: On the INT-1 fixture (Appendix DATASETS), measured correct-assignments fraction ≥ 0.99999999.

   Verification: Test: replay INT-1 with ground-truth mapping; compute fraction; assert bound.

2. Must **REQ-INT-02 – Loss/duplication bound.** Under the same conditions, lost or duplicated hits shall be $\leq 0.001\%$ of total hits.

   Acceptance: Counters and reconciliation against ground-truth yield $\leq 1 \times 10^{-5}$ fraction lost+dup.

   Verification: Test: reconcile emitted hits vs. truth; assert bound.

3. Must **REQ-INT-03 – Sorter persistence latency.** The sorter shall persist each mapped hit within $0.5\,\mu s$ (median) of the detector-provided timestamp, measured per Appendix PERF-TIMEBASE.

   Acceptance: Median persist delay $\leq 0.5\,\mu s$ on INT-1; P95 reported for visibility.

   Verification: Test: align timebases; measure per-hit delay; assert median bound.

4. Must **REQ-INT-04 – Per-second telemetry.** While acquiring, the system shall publish per-second counters of total hits, lost hits, mis-mapped hits, and median latency via the telemetry bus.

   Acceptance: Telemetry stream present at 1 Hz; fields populated; values match internal counters.

   Verification: Test: subscribe; compare samples to internal metrics.

3.1.11 UI Shell

This section describes the UI counterpart of the feature only. Every UI component is backed by a non-UI component and its business logic is tested away from the UI in the form of unit/integration/end-to-end testing.

1. Must **REQ-UI-01 – Standalone launch.** The application shall launch as a standalone process when the user double-clicks the 4D-STEM UI icon or selects it from the native OS menu.

   Acceptance: Main window becomes interactive; a single primary process is visible in the OS process list.

2. Must **REQ-UI-02 – Restricted detector controls.** The UI shall display only the detector controls listed in Appendix UI-A1 (whitelist).

   Acceptance: Across all screens and states, only whitelisted controls appear; no navigation path exposes non-whitelisted controls.

3. Must **REQ-UI-03 – Derived detector settings.** Where specified in Appendix UI-A2, detector settings shall be derived automatically from scan parameters without additional user interaction. If the setting can be abstracted from the user, then it should be.

   Acceptance: Changing any covered scan parameter updates the mapped detector setting. The change is reflected in serval

4. Must **REQ-UI-04 – Scan engine selector.** The UI shall present a control to select which scan engine is in use, always visible. Supported scan box units are present in the drop down list but disabled if not found.

   Acceptance: Control is visible without opening dialogs; selection persists to the active measurement context.

5. Must **REQ-UI-05 – Scan geometry controls.** The UI shall present controls for geometry including: scan height and width (in probe positions), prescan x/y, always visible.

   Acceptance: Editing values updates the pending scan geometry such that they take effect when a stem acquisition is run.

6. Must **REQ-UI-06 – Dwell control.** The UI shall present a dwell-time control, always visible.

   Acceptance: Value changes propagate to pending acquisition parameters and appear in the run summary / logs.

7. Must **REQ-UI-07 – Z-frames control.** The UI shall present a control for the number of frames Z (frame = one full scan of $H \times W$), always visible.

   Acceptance: Z appears in the run summary and is written to measurement metadata.

8. Must **REQ-UI-08 – Start (Preview).** The UI shall provide a Start (Search/Preview) control.

   Acceptance: Activating enters Preview mode and renders live vADF; control indicates running state.

9. Must **REQ-UI-09 – Start (Acquire).** The UI shall provide a Start (Acquire) control.

   Acceptance: Activating starts acquisition with current parameters and writes data to the selected output format(s).

10. Must **REQ-UI-10 – Output Location.** The UI presents a widget to collect output location. The widget accepts text input and selection from the native filesystem widget

    Acceptance: Acquisition puts the data and metadata into this location

11. Must **REQ-UI-11 – Output format.** A dropdown containing supported output formats is presented and always visible.

    Acceptance: Output format is a field in the metadata. After acquisition the data shall be in the requested location in the required format

12. Must **REQ-UI-12 – Workspaces.** Users shall be able to create/open a workspace; a workspace can contain multiple measurements.

    Acceptance: New/open actions work; workspace path is stored in recent list; multiple measurements list correctly.

13. Must **REQ-UI-13 – Measurement metadata.** Each measurement shall have metadata persisted as JSON (schema defined in Data Formats), accessible via the UI but hidden by default. All scan and detector parameters are included in the measurement metadata plus comments written by users themselves.

    Acceptance: 'Show metadata' displays a measurement viewer, hidden by default; file exists on disk with schema version and passes schema validation.

14. Must **REQ-UI-14 – Stop button.** The UI shall provide a Stop control that is enabled only while Preview or Acquire is running.

    Acceptance: Disabled when idle; enabled within 200 ms of run start; returns UI to idle on activation.

15. Must **REQ-UI-15 – Stop semantics.** When the user presses *Stop*, the system shall finish the current probe position and then stop. A preference 'Stop at frame boundary' shall, when enabled, finish the current frame before stopping.

    Acceptance: Default behavior ends within one probe after Stop; with the preference enabled, the run ends at the next frame boundary; manifests reflect a clean final frame state.

    Verification: Test: trigger Stop mid-frame with option off/on; verify end position and manifest entries.

16. Must **REQ-UI-16 – ROIs.** The user shall be able to create, update/move and remove rectangular, circular or annular ROIs by click-and-drag on the viewport.

    Acceptance: ROI appears with handles; coordinates shown in status.

17. Must **REQ-UI-17 – ROI to scan geometry.** The system shall translate a rectangular ROI into an equivalent scan geometry. ROIs snap to the pixel grid so values derived are integers.

    Acceptance: Computed H and W equal ROI extents; pending scan parameters update accordingly.

18. Must **REQ-UI-18 – ROI for diffraction views.** The system shall allow using ROI selections to build diffraction images (e.g. averaged CBED of the selection).

    Acceptance: Selecting a rectangular ROI automatically computed the average CBED

19. Could **REQ-UI-19 – ROI expanded view.** A ROI shall be expandable via invoking the 'right click ¿ Expand ROI' function. The system produces a full size, zoomed in image from the selection and optionally enables updating the scan geometry to reflect the new sub-region

    Acceptance: Right-click ¿ Expand ROI opens the new view of the current image. The new view has the correct bounds and scan is updated.

20. Could **REQ-UI-20 – Dockable controls.** Control panels shall be dockable/undockable and toggleable via the View menu. A default layout is exposed for return to factory settings.

    Acceptance: Panels can be moved, floated, and toggled.

21. Must **REQ-UI-21 – Persistence.** UI settings (layout, last workspace, panel visibility, scan control values) shall persist across sessions and at the granularity of a workspace. Detector specific controls persist per workstation, rather than per workspace.

    Acceptance: Close/reopen restores prior state on the same machine/user.

22. Must **REQ-UI-22 – Connectivity loss handling.** On loss of connectivity to Serval, scan engine, or detector, the UI shall display status and attempt reconnect every 1 s until success.

    Acceptance: Observed retry interval for at least 10 min; status banner visible.

23. Must **REQ-UI-23 – Startup without connectivity.** If no connectivity is available at startup, the UI shall present a non-blocking status and allow offline configuration. Any setting or control relating to the unconnected device or server shall be blocked when not connected.

    Acceptance: Main window is usable; status banner visible; configuration pages accessible. Scan controls disabled when no scan generator available. Controls that drive Serval are disabled when serval is not available.

24. Must **REQ-UI-24 – Logging.** The UI shall write a log of important events to disk and to console; log level is controlled by environment variable `LOG_LEVEL`. Log location is user-configurable under `File > Preferences` but defaults to within the workspace configuration.

    Acceptance: Different `LOG_LEVEL` values change verbosity; files created at default paths; preference overrides path; Serval redirection present.

25. Must **REQ-UI-25 – Disk space guard (preflight + warning).** On *Start (Acquire)*, the system shall evaluate free space on the selected target volume. If free space is at or below the near-full threshold (default 5% of the volume), the system shall present a low-space warning before starting. If the volume reports 0 writable bytes free, the acquisition shall not start.

    Acceptance: On pressing *Start (Acquire)*: (a) if free space ¿ threshold, acquisition begins; (b) if free space $\leq$ threshold, a low-space warning is shown and the user may proceed; (c) if free space $= 0$, acquisition is blocked and an error is shown. All cases are recorded in the log.

26. Must reqDetector info and health available UI shall display from view ¿ detector health a UI panel describing the detectors information from serval The UI displays accurate information

3.2 Non-Functional Requirements

3.3 Legal and Regulatory Requirements

Licenses. What can we do? What can't we do?

4 Appendices

4.1 Glossary

4.2 Use Cases and Diagrams