

The VALIS CLI Manual

VALIS-REF-001

TITLE	The VALIS CLI Manual
DOCUMENT	VALIS-REF-001
STATE	Adopted, first issue, October 21, 2026
EVIDENCE GRADE	E1 Evidenced. The manual is generated in part from the canonical source it describes, and every statement that compares the source against the shipped preview package was computed from the two artifacts at drafting, not asserted from memory. The reconciliation section prints those computations and their results
EDITION	First issue
DIGEST, SHA-256	d75f923cc5033b42bdbf73ade59676b4d13d4697038a2bbf8cac807b2cd025da
LICENSE	CC BY 4.0, attribution the only condition; code samples, Apache License 2.0

This export was generated mechanically from the canonical Markdown source of record. Its digest stands above and in the Library registry, and the Markdown export of this document carries that digest exactly.

The VALIS CLI Manual

Cover block

Series and number: REF, VALIS-REF-001 Series mark: VALIS Title: The VALIS CLI Manual State: Adopted, first issue, October 21, 2026 Adoption act: Founder Ricky, October 21, 2026, 6:41 PM PDT, by the words "you have my approval for the CLI manual and then for all the documentation so that you can publish that into its corresponding places repo website app CLI"; the CLI manual is the first document that ruling names, and the adoption is recorded in the corpus registry Kind: reference Evidence grade: E1 Evidenced. The manual is generated in part from the canonical source it describes, and every statement that compares the source against the shipped preview package was computed from the two artifacts at drafting, not asserted from memory. The reconciliation section prints those computations and their results Record age: the source facts in this manual were verified against the canonical CLI tree and the shipped preview package of record on October 21, 2026. The shipped preview predates the WO-58 CLI update, which stands in flight; where the update changes a fact this manual prints, the carry-forward section says so and an errata entry or reissue follows the update's landing Digest: computed at filing and carried in the corpus registry; this document prints no digest of itself License line: the text of this document is licensed under CC BY 4.0, attribution the only condition; code samples in this document are under the Apache License 2.0; the ARC and VALIS names and marks are not licensed under either instrument. Decided by Founder Ricky on October 21, 2026, at 6:45 PM PDT Next review date: October 21, 2028, on the Manuals review clock of two years the charter sets, and at every CLI release that changes a fact printed here, whichever comes first Stable address: the corpus registry entry for VALIS-REF-001 Authority note: this manual describes a program. It confers no authority, admits nothing, and settles nothing. Under the charter's fourth standard, a document describes; governance disposes. Where the manual describes the governed boundary, it describes a refusal posture, and the description is not a path around it

Make a Statement!

How to read this manual

This manual describes one program, the ARC workstation command line, in two builds that must never be confused. The source of record is the canonical tree in the programme's build home, package arc-foundation-cli, whose version the package itself states in one place. The shipped preview is the downloadable package of record, arc-cli-preview.zip, build stamp 3f339049, which is what an operator actually installs today. The two builds differ, and the differences are listed in the reconciliation section rather than smoothed over. Every section of this manual describes the source of record unless it says otherwise, and says so where the shipped preview behaves differently.

The manual's reference core is the CLI's own built-in documentation, the fourteen addressed sections the program prints under arc docs, promoted into print under the corpus charter. Those sections are the program describing itself, and this manual carries them at full strength, then surrounds them with what a reference owes its reader: how to install and verify the program, how its faces behave, what files and variables it honors, and exactly where its two builds disagree.

The surface

The program is a Python application, run as arc, that gives a person and their intelligence a field to work over: a project of relations on their own machine, a language for asking about it with the evidence shown, one write path that materializes derivations inside the project, and the node machinery of registration, handshake, sharing, presence, discovery, and sealed messaging. Nothing in it reaches a governed effect. Its boundary section, printed in full below, states the posture in the program's own words.

Install and verify

The shipped preview is the installable artifact. It is verified the same way every artifact in this programme is verified: against its published digest, before it runs. The digest of the shipped preview of record is sha256
d903674e48c69856a073593100d4e68900aa218badea5f5fad59d31c3d6c19fa at
240,108 bytes, and the sums file published with the downloads is signed. An operator who cannot verify the package does not install it. The source of record runs from its own tree under its project interpreter, and a build is stamped at packaging time with the commit that carries the packaged code; stamping is a packaging act, and the stamp names provenance, never quality.

The fourteen sections

The program's built-in documentation stands in fourteen addressed sections, printed by arc docs with no argument as an index and by arc docs with a topic name one at a time. The sections follow, each under its own heading, in the program's words.

Start

Topic name: start

arc init creates a project, or arc db init initializes a field in the current directory.

Shell

Topic name: shell

arc repl starts the interactive shell, with history and Tab completion. arc home redraws the opening frame, explore prints the example field, language lists the language's openings, config shows the boundary posture, and exit or quit leaves the shell. Ctrl+C at the prompt leaves the shell, as the opening frame's footer states; Ctrl+C during a command stops that command, and the shell goes on.

Compute

Topic name: compute

arc query runs the ARC Language, read-only. Two forms are supported, the keyword DSL (EXPOSURE OF ORDER-1) and the founder form (from agent where role == 'operator' select id, authority, constraints).

Derive

Topic name: derive

arc reify "" as materializes a derivation as a new persisted field version. It is the only write path, and it stays inside the project.

Programs

Topic name: programs

arc run executes a statement program, one query, select, or reify statement per line.

See

Topic name: see

arc graph visualizes relations, arc field draws the relational field itself with per-entity metrics, arc db shows field state, arc evidence shows the evidence log, and arc inspect examines provenance for one target.

Intelligence

Topic name: intelligence

arc connect binds the intelligence you already use, a provider endpoint or a local model, to the project. From then on, ask in your own words. ARC shows the exact ARC Language query before running it and answers with the evidence. The binding proposes language only; it never settles, approves, or executes an effect.

Register

Topic name: register

arc register prepares this node's signed registration proposal in its node home (~/.arc-node by default). The node's key is born there and never leaves the machine. The proposal reaches the registry through a door, the registry settles it, and an admitted node is listed and receives a Statement. arc register status checks the public registry feed. Listing is a statement about the node, not a connection to anything.

Handshake

Topic name: handshake

arc handshake serve runs this node's side of the federation handshake (discovery, signed capability advertisement, version and suite negotiation, identity and revocation checks, trust-policy acceptance) using the sealed protocol server in the node home's federation directory. arc handshake connect --peer runs the five steps against a serving peer. Trust is granted only under this node's own policy or its operator's explicit confirmation, and each side keeps the signed receipts. A handshake establishes a channel and recorded trust only; it settles nothing and grants no permission for consequential actions.

Share

Topic name: share

arc share export --store writes one signed bundle holding the store's current field, its version and digest chain head, and this node's address. The operator carries the file. arc share receive --bundle --store verifies the signature, the content digest, and the sender's standing in this node's handshake trust, then imports the records as a new version of the receiving store, under the receiver's own writer. An untrusted sender is refused unless --accept-untrusted accepts that one bundle from that named sender. Sharing is a bounded act, never a synchronization, and connection alone never copies a field.

Presence

Topic name: presence

arc presence publish signs one heartbeat from this node's key in its node home and writes it to /presence/beat.json, appending the same signed beat to the heartbeats.jsonl log beside it, the lane's own shape and signature discipline, so any checker built for the founding lane reads it. The beat carries the node's address, its name when known, the time, its posture, and a counter that only moves forward. arc presence watch repeats the beat on an interval, and after each beat the watch pushes it outbound to the node's configured relays (the foundation relay by default; the list lives in /presence/relays.json) for the registry's publisher to collect and verify. The local signed log in the node home remains the source of truth, and nothing is fetched from the node. arc presence status --address reads a beat, a beat log, or the registry's published presence

feed from a file or URL, verifies the signatures against the key the address itself commits to, and derives the state: online within 20 minutes, stale within 75, offline beyond, unknown when nothing verifies. A node's own assertion is never the evidence; the signature is.

Discover

Topic name: discover

arc discover reads the network's two published records, the registry's listing and the presence feed, and joins them by address: every listed node with its operator, roles, and locality, its operator class and custodian of record where the record names them, and the presence state the feed's record supports for it, derived from the recorded last seen on the lane's windows. A node appears only because its registration settled and the registry listed it. Discovery scans nothing, probes no endpoint, and cannot find an unlisted node. Listing is a published statement about a node, not a connection to it; reaching a node is the handshake, a separate act. arc discover --name resolves one listed node by the operator name or node name it gave, and returns its address ready for the message flow.

Mailbox

Topic name: mailbox

arc mailbox deposit seals one message to a named recipient's key and leaves the envelope in a mailbox the operator names. The mailbox takes two forms. A directory either node can reach: a synced folder, a mounted share, a git-backed folder. Or a relay, a small service that carries sealed envelopes over HTTP for nodes that share no directory; the relay holds ciphertext only, grants no trust, and releases envelopes only against a collection request signed by the recipient's federation key. --relay names one, ARC_MAILBOX_RELAY_URL or the node home's mailbox-config.json configures one, and with no address named the documented institution relay is the default; any operator can run a relay and point a node at it, because a relay that cannot be replaced would hold a position ARC grants to no one. arc mailbox collect opens the envelopes addressed to this node, verifies each sender's signature and standing in this node's handshake trust, and delivers the messages into the node home. Either transport holds ciphertext only; neither ever holds field data or a key. A mailbox moves messages only: it grants no authority and settles nothing. Envelopes are bounded in payload size, in number waiting per recipient, and in lifetime, set by --ttl-hours or by a class (urgent 24 hours, standard 168, durable 2160), and an expired envelope is reported and swept, never delivered.

Boundary

Topic name: boundary

<_ enters governed settlement for an exact proposal file. No KORE adapter is connected in this build, so the boundary refuses. Settlement is not execution.

The command index

The program's help registry carries 41 entries, and the grouped help an operator sees under arc help is derived from that registry, so a new command joins both by being entered once. The registry follows, generated from the source of record at drafting.

- <_: Validate an exact proposal (refuses without KORE adapter)
- config: Environment and boundaries
- connect: Bind an intelligence to this project (yours, any platform)
- database: Inspect local database state
- db: Inspect local database state
- db init: Initialize a native field in the current directory
- discover: Discover the network's listed nodes, and resolve a name to its address
- docs: Read the built-in documentation
- evidence: Evidence and verifier posture
- exit: Leave ARC
- explore: Explore example relational field
- field: Draw the relational field itself, with per-entity metrics
- graph: Visualize the relations in the current field
- handshake: Federation handshake: serve this node, or connect to a peer
- handshake connect: Run the five handshake steps against a serving peer
- handshake identity: Show or create this node's federation identity
- handshake serve: Run this node's side of the federation handshake
- help: Show commands
- home: Return to opening screen
- init: Initialize a project with a persisted native field
- inspect: Examine synthetic relational evidence
- lang: Show ARC Language examples
- language: Show ARC Language examples
- mailbox: Send and receive sealed messages between nodes
- mailbox collect: Open the envelopes waiting for this node in the mailbox or at a relay
- mailbox deposit: Seal one message to a recipient and leave it in the mailbox or with a relay
- presence: Publish and check signed node presence
- presence publish: Sign one presence beat from this node
- presence status: Verify a node's presence from its published beats
- presence watch: Keep signing presence beats on an interval
- query: Run a read-only query (DSL or founder form)
- quit: Leave ARC
- register: Propose this node's registration to the ARC Network
- register status: Check whether this node is listed in the public registry

- reify: Materialize a query derivation as a new field version
- repl: Start the ARC REPL (this shell)
- run: Execute an .arc statement program against a project field
- settle: Validate an exact proposal (refuses without KORE adapter)
- share: Share field content with another node as a signed bundle
- share export: Export this store's current field as a signed share bundle
- share receive: Verify a share bundle and import it as a new version

Ask, the intelligence binding

arc connect binds the intelligence the operator already uses to the project. From then on the operator asks in their own words. The program shows the exact ARC Language query before running it and answers with the evidence the query returned. The binding proposes language only; it never settles, approves, or executes an effect. In the source of record, the provider invocation that carries the ask passes the provider's non-interactive execution mode with its repository trust check skipped, a one-line fold the WO-58 update carries, because the ask runs inside the operator's own project on the operator's own machine. The shipped preview does not carry that fold, and an ask that reaches the provider's repository check on the preview build can refuse where the source build proceeds. The fold is listed in the reconciliation, and the binding's end-to-end behavior over a live project field stands proven on the founder's own machine under WO-58 item 14.

The home face and the version

The program's opening face reports the installation, the identity posture, and the field. In the source of record the face also surfaces the live record: the registry listing and the presence feed joined by address, the same join arc discover performs, so the opening state of the program is the network as published, and where the network is unreachable the face says so rather than inventing a state. The version the program prints derives from one source of truth, the package version, which stands at 0.2.1-dev in the source of record. The shipped preview prints a hardcoded label, 0.2.0-dev, from four places in its code, while its own package metadata says 0.2.1.dev0, so on the preview build the program and its package disagree about the program's version. That disagreement is a defect of record, repaired in the source of record, and it is listed in the reconciliation rather than corrected silently here.

Color

The program renders its wordmark and headers in gold. In the source of record the palette is chosen against the terminal's actual capability: terminals that advertise truecolor receive the truecolor golds, and terminals that do not, including a 256-color profile such as a stock Mac Terminal Basic profile, receive palette-safe golds from the 256-color cube, because the truecolor golds render as red where truecolor is

unavailable. The shipped preview carries the truecolor literals only. The repair is in the WO-58 update scope and in the source of record.

Files, stores, and environment

A project field lives in the project the operator names. The program reads its environment in a handful of named places, computed across the whole package at drafting: ARC_PROJECT names the project, in project.py; ARC_NODE_HOME names the node home, in handshake.py; ARC_CONFIG_DIR names the configuration directory the intelligence binding reads, in intelligence.py; ARC_FIELD_KIT_DIR locates the field kit for the share bridge, and ARC_MAILBOX_DIR names a mailbox directory, both named in app.py; ARC_MAILBOX_RELAY_URL configures a mailbox relay, as the mailbox section above states; and COLORTERM then TERM decide the palette, as the color section states The node home, ARC_NODE_HOME or ~/.arc-node by default, holds the node's key, which is born there and never leaves the machine, its federation material under its own folder, its presence log, and its mailbox configuration. A store is a file; its digest chain is its history; the evidence log is the record of what was asked and answered.

Reconciliation, the source against the shipped preview

Every row of this table was computed at drafting from the two artifacts themselves. A row that disagrees is a defect or a pending press, and each names which.

- Version, source against preview: the source package states 0.2.1-dev in one place and every printed label derives from it. The preview prints 0.2.0-dev from 4 hardcoded sites and states 0 occurrences of the source version. Disagreement, repaired in source, awaiting the WO-58 press.
- Package metadata against program label, preview internal: the preview's own package metadata in its tree states 0.2.1.dev0 in the source pyproject, while the preview program prints 0.2.0-dev. Disagreement of record.
- Built-in documentation: the preview carries the fourteen sections (present in its app code), so arc docs agrees across the two builds in structure; the manual's remaining rows are where they part.
- The ask fold: the source carries the provider repository-check skip (present); the preview does not (absent). Disagreement, folded into source under WO-58, awaiting the same press.
- The home face live record and the palette-safe golds stand in the source of record under the WO-58 update and not in the preview of stamp 3f339049. Awaiting the same press.
- Build stamp: the preview's stamp names commit 3f339049, and the preview archive carries 70 entries. The stamp is provenance, and no row of this table upgrades it into a claim about the source, which has moved since.

Carry-forward

One open item rides this edition. The WO-58 CLI update, one build pressed once, carrying the unified version, the live-record home face, the palette-safe golds, and the ask fold, stands in flight at this manual's adoption. This manual describes the source of record, which already carries those changes, and the shipped preview, which does not yet. When the update's press lands and the updated build is installed and verified on the seats it targets, this manual's reconciliation is re-run against the new package of record, and the manual is reissued or errata'd from that computation. Until then, an operator reading this manual beside the preview build should read the reconciliation as the map of their differences.

What this document does not establish

This manual does not establish that the shipped preview and the source of record agree. They do not, and the reconciliation prints the disagreements rather than establishing a unity that has not landed.

This manual is not a specification of a future product. An earlier draft CLI specification, which stated on its face that it specified a product that does not yet ship, stands retired into a labeled design record under the corpus charter and is not a source for this manual or for the program.

This manual does not establish a governed path. The program's boundary refuses governed settlement in this build, and nothing described here is a route around that refusal.

This manual does not establish the program's behavior on a machine it has not run on. The ask binding stands proven on the founder's machine at the scope WO-58 records, and the manual claims no wider device proof.

Drafted and adopted October 21, 2026, by Noetica, Keeper of the Record, under WO-56. Sources of record: the canonical CLI tree in the programme's build home, its package metadata, its built-in documentation and help registry, and its README; the shipped preview package of record, arc-cli-preview.zip at build stamp 3f339049, read directly for every comparison this manual prints; and the corpus charter, ARC-GOV-001, whose VALIS-REF-001 entry commissions this manual and fixes its method.