Network Working Group
Internet-Draft
Intended status: Informational
Expires: 22 November 2026

B. Morrison
Alter Meridian Pty Ltd (~truealter)

May 2026

An Agent-Channel Frame for Identity-Keyed Fan-Out Delivery to Concurrent Sessions

draft-morrison-agent-channel-fan-out-00

Abstract

This memo specifies an application-layer frame format and a delivery model by which the several concurrent agentic sessions of a single identity-bound principal, and the recognised members of an organisational identity substrate, exchange short structured messages. The frame, termed the agent-channel frame, is a transport envelope: it carries a closed-catalogue kind discriminator, a structured per-kind payload, an identity attribution pair, and an inline provenance block. Delivery is fan-out: a sender names a recipient scope rather than a single endpoint, and the scope is expanded at delivery time against the recipient's subscriptions. Recipients receive frames over a per-handle Server-Sent Events stream and MAY narrow what they receive with a subscribe-time filter expression. Frames are ephemeral routing units; the memo specifies only the wire envelope, the scope-expansion grammar, the subscribe filter grammar, and the delivery semantics. Frame persistence, where an implementation chooses to retain frames for replay, is out of scope and is not specified. The memo composes with the handle namespace of [IDPRONOUNS], the discovery surface of [MCPDNS], and the cross-organisational ceremony of [IDACCORD]; no new transport and no new handle category is introduced.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 2 November 2026.

Table of Contents

1. Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

2. Introduction

A single human principal increasingly operates several artificial-intelligence agent runtimes at once: multiple concurrent sessions of one coding assistant, a separate assistant in an integrated development environment, a command-line agent, and a background runtime daemon. Those surfaces need to exchange short messages with one another: to advise a sibling that a file is being edited, to hand

work over when one session ends, to ask a question and collect answers, to announce that a branch was merged.

In current practice such messages have no shared transport. Where two sessions of the same principal must coordinate, the prevailing fallback is a file written to a shared temporary directory and a manual instruction to a second session to read it. That fallback has no addressing, no delivery guarantee, no attribution, and no filtering; it does not reach a session on a second host, and it does not reach the principal's runtimes operating under a second organisational context.

This memo specifies a frame format and a delivery model that replace the file fallback. The contribution is deliberately narrow. The companion memo [SUBSTRATE] argues that _coordination_ (the deconfliction of conflicting action) should not be standardised as an envelope protocol, because envelope coordination re-centralises an inherently distributed problem. The present memo does not contradict that argument. It specifies a _delivery_ frame for messages a session has _chosen_ to send: an advisory, a handover, a question. It is a transport, not a coordination protocol. No frame in this memo's catalogue compels a recipient to any action; no frame carries a lock a recipient must honour; the lock-shaped frames of Section 5 are advisory announcements, and a recipient that ignores one remains conformant. Reconciliation of conflicting action, where it occurs, occurs through the substrate-observation cascade of [SUBSTRATE], not through this memo's frames.

The frame is identity-keyed. A sender names a recipient _scope_: "all my sessions", "all sessions of one tool", "one named session", or "all members of an organisation"; the scope is expanded against recipients' live subscriptions at delivery time. The fan-out is the load-bearing delivery primitive: a sender addresses a set, not an endpoint, and the set is resolved by the delivery substrate from identity, not from a registry of endpoints the sender maintains.

The frame is ephemeral. A delivered frame is a routing unit with a time-to-live; once delivered, the delivery substrate is under no obligation specified by this memo to retain it. Where an implementation retains delivered frames so that a late-joining session can replay recent traffic, the retention mechanism, its ordering properties, and its retention horizon are implementation matters outside the scope of this specification. This memo specifies the wire envelope and the delivery semantics; it does not specify a store.

3. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 RFC2119 [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined for the purposes of this document. Terms previously defined by the referenced Morrison-family memos retain their established meaning and are reproduced here only when operative for the present specification.

~handle A principal identity handle as defined by [IDPRONOUNS]. A Sovereign-tier handle is human-controlled (e.g. ~alice); an Instrument-tier handle is agent-runtime-vendor-controlled and conventionally prefixed ~cc- (e.g. ~cc-example-model).

Agent-channel frame The application-layer message unit specified by this memo. A single JSON object [RFC8259] comprising an envelope, a kind discriminator, a structured payload, an identity attribution pair, and a provenance block.

Kind The closed-catalogue discriminator naming the frame’s purpose. Section 4 enumerates the catalogue. An implementation that receives a kind outside the catalogue MUST reject the frame.

Payload The kind-specific structured body of a frame. Each kind has exactly one payload shape; a frame whose payload shape does not match its kind is malformed.

Recipient scope A string naming the set of recipients a frame is to be delivered to. Section 6 specifies the scope grammar. A recipient scope is not an endpoint; it is expanded against live subscriptions at delivery time.

Delivery substrate The network-addressable system that accepts a submitted frame, expands its recipient scope, and emits the frame to the subscriptions the scope names. The delivery substrate is addressed per the discovery mechanism of [MCPDNS].

Per-handle stream A single Server-Sent Events HTML-SSE [RFC8441] stream associated with one ~handle, over which the delivery substrate emits the frames whose expanded recipient scope includes a session of that handle.

Subscribe filter An expression supplied by a subscriber at the time

it opens its per-handle stream, narrowing the frames the delivery substrate emits to that subscriber. Section 8 specifies the filter grammar.

Time-to-live A per-frame integer, in milliseconds, after which a sender declares the frame's content no longer operative. The time-to-live is advisory to recipients; it is not a retention directive to the delivery substrate.

4. The Agent-Channel Frame

An agent-channel frame is a single JSON object [RFC8259]. Its fields are specified below. An implementation receiving a frame MUST reject it if a REQUIRED field is absent, if any field carries a value outside the constraints below, or if an unrecognised field is present (the frame object does not admit extension fields; see Section 12).

4.1. Envelope Fields

envelope_version (string, REQUIRED) The wire-contract version. This memo specifies version "1.0". A receiver that does not implement the stated version MUST reject the frame rather than attempt a partial parse.

frame_id (string, REQUIRED) A version-4 UUID generated by the sender, unique to this frame. Receivers MAY use frame_id to discard a duplicate delivery of the same frame.

kind (string, REQUIRED) One of the fifteen catalogue values of Section 4. A kind outside the catalogue MUST cause rejection (Section 10).

sender_handle (string, REQUIRED) The ~handle of the sending principal, in the canonical handle form of [IDPRONOUNS]. The delivery substrate MUST verify that sender_handle matches the authenticated identity of the submitting session and MUST reject a frame whose sender_handle does not.

recipient_handle (string, REQUIRED) The ~handle of the target principal, in canonical handle form. For a frame addressed to the sender's own sessions, sender_handle and recipient_handle are identical. The recipient _scope_ (Section 6) further qualifies which sessions of recipient_handle receive the frame; recipient_handle alone identifies the target principal.

created_at (string, REQUIRED) An RFC 3339 timestamp, with time zone,

at which the sender emitted the frame. The delivery substrate MAY reject a frame whose created_at lies implausibly far in the future or the past relative to substrate time.

ttl_ms (integer, OPTIONAL) The frame's advisory time-to-live in milliseconds. When absent, the recipient applies an implementation-default horizon. The field is advisory to the recipient (it indicates when the sender considers the content stale); it places no retention obligation on the delivery substrate.

payload (object, REQUIRED) The kind-specific body, with the shape specified for the frame's kind in Section 5.

4.2. Attribution Fields

Every frame carries the tier-structured attribution pair of [IDCOMMITS], lifted from version-control commit time to wire time.

acted_by (string, REQUIRED) The Sovereign-tier ~handle of the human principal on whose behalf the frame is sent.

drafted_with (string, REQUIRED) The Instrument-tier ~handle of the agent runtime that composed the frame.

The attribution pair lets a receiver distinguish a frame a human authored directly from a frame an agent runtime composed, without a side-channel lookup. An agent-channel frame and an Acted-By: / Drafted-With: commit trailer block carry the same attribution shape: one at message time, one at commit time.

4.3. Provenance Block

Every frame carries an inline provenance block so a receiver can assess the frame's basis without an out-of-band query. The block comprises five fields.

provenance_compute_location (string, REQUIRED) Where the inference behind the frame's content was computed. One of "server-active", "server-aggregate", or "local-only". The value "local-only" asserts that the frame's content derives from inference that did not leave the principal's device.

provenance_method (array of strings, REQUIRED, non-empty) One or more derivation-method tags identifying how the frame's content was produced (for example, a session-context snapshot, a pre-compaction snapshot). At least one tag is REQUIRED.

provenance_return_ref (string, OPTIONAL) A reference to a return record, where the frame’s content is derived from a recipient’s data and the implementation maintains a return-accounting surface. Absent where inapplicable.

provenance_context_check (string, REQUIRED) One of "passed" or "skipped". The value "passed" asserts that the sender confirmed the frame is not emitted in a context the sender’s policy prohibits; "skipped" asserts the check is inapplicable to the frame.

provenance_basis (string, REQUIRED) A short tag naming the stream the frame belongs to and the basis under which that stream emits.

The provenance block is descriptive metadata carried for the receiver’s benefit. This memo does not specify a conformance behaviour keyed to any provenance field; an implementation MAY use the block to inform its own acceptance policy.

5. The Frame-Kind Catalogue

The kind field takes one of exactly fifteen values. The catalogue is closed: a sixteenth value is an error (Section 10), not an extension point. The catalogue is grouped below by function; the grouping is descriptive and is not encoded on the wire.

5.1. Advisory and Broadcast Kinds

agent_advisory A session announces what it is doing, for example that it is about to edit a named file, or that it is operating in a named working tree on a named branch. Default recipient scope is the sender’s own sessions.

agent_broadcast A session announces a completed event other sessions should know of, for example that a branch was merged, or that a sibling’s working tree is now stale. Carries a coarse event class so a subscriber may filter without parsing the body.

5.2. Handover Kind

agent_handover A session passes work to another session, carrying the prose content a principal would otherwise place in a shared temporary file and a list of pointer references (file paths, change-request references, working-tree paths). This kind replaces the temporary-file fallback described in Section 1.

5.3. Lock-Shaped Kinds

The following three kinds are _advisory announcements_. They announce a session's intent regarding a named resource. They do not, and cannot, compel a recipient. A recipient that receives an agent_lock_request and proceeds to act on the named resource anyway is conformant; the announcement is information, not authority. Enforcement of mutual exclusion, where an implementation requires it, is performed locally by the operating environment (for example, a filesystem advisory lock), not by this memo's frames.

agent_lock_request Announces that the sender claims a named resource for an advisory lease, carrying a lease identifier and a lease lifetime.

agent_lock_release Announces that the sender has released an advisory lease, carrying the lease identifier.

agent_lease_extend Announces that the sender extends the lifetime of an advisory lease, carrying the lease identifier and the additional lifetime.

The lease identifier is a correlation value only. The delivery substrate MUST NOT treat it as a key into any lock table, MUST NOT enforce uniqueness of an outstanding lease, and MUST NOT reject an agent_lock_release whose lease identifier does not match a prior agent_lock_request. Correlation of the three kinds is performed by recipients; the delivery substrate validates only their shape.

5.4. Question-and-Answer Kinds

agent_query Asks a question and names a scope to which answers should be delivered, carrying a query identifier and a caller-side collection window.

agent_response Answers an agent_query, carrying the query identifier it answers and the responding session's identifier.

agent_return_event Records the closure of a question-and-answer exchange against a return-accounting surface, where the implementation maintains one, carrying a return reference and an optional query identifier.

5.5. Decision-Request Kind

agent_binding_moment Carries a structured decision request from one session to another. Its payload is the eight-slot grammar of Section 5.10.

5.6. Diagnostic Kinds

peer_diagnostic_request Reports an observed symptom to a peer for troubleshooting, carrying a diagnostic identifier and a coarse severity.

peer_diagnostic_response Answers a peer_diagnostic_request, carrying the diagnostic identifier and a finding.

5.7. Convergence Kinds

intent_declare Announces a session’s intent to incur a class of costly external effect, carrying a namespaced class identifier, a pointer to the intended effect, and a decay horizon.

intent_withdraw Withdraws a previously declared intent before its effect is executed, carrying the class identifier and a pointer to the declared intent.

flush_executed Announces that a session has executed a batched effect on behalf of a set of declared intents, carrying a pointer to the result and the pointers of the included intents.

The three convergence kinds let a set of sessions batch a costly effect (for example, opening a single change request for several pooled changes) without electing a coordinator: each session announces its intent, observes the others’ announcements, and one session executes the batched effect. As with the lock-shaped kinds, the announcements carry no authority; a session that declares an intent and a session that executes a flush each act on their own observation of the announced traffic.

6. Payload Schemas

Each kind has exactly one payload shape. A frame whose payload does not satisfy the shape for its kind is malformed and MUST be rejected. The schemas below give the REQUIRED and OPTIONAL fields of each payload; string-length and numeric bounds are stated where an implementation is expected to enforce them.

6.1. agent_advisory

advisory_text (string, REQUIRED, 1 to 2048 octets) What the sender is doing.

file_refs (array of strings, OPTIONAL) Files the advisory concerns.

worktree (string, OPTIONAL, up to 512 octets) The working-tree path

the sender operates in.

branch (string, OPTIONAL, up to 256 octets) The branch the sender is on.

6.2. agent_broadcast

broadcast_text (string, REQUIRED, 1 to 2048 octets) The announcement body.

event_class (string, REQUIRED) One of "merged", "stale", "released", "other". A coarse class permitting subscriber-side filtering without body parsing.

refs (array of strings, OPTIONAL) Branch, change-request, or commit references the broadcast concerns.

6.3. agent_handover

previous_session_id (string, REQUIRED, 1 to 128 octets) An opaque identifier for the handing-over session.

next_session_id (string, OPTIONAL, up to 128 octets) An identifier for the session that claims the handover. Absent at emission; a claiming session populates it by a subsequent frame.

handover_body (string, REQUIRED) The handover content.

pointer_refs (array of strings, OPTIONAL) File paths, change-request references, decision identifiers, and working-tree paths the handover points at.

6.4. agent_lock_request

resource (string, REQUIRED, 1 to 512 octets) The resource the sender claims an advisory lease over.

lease_id (string, REQUIRED) A version-4 UUID correlating this announcement with later agent_lock_release and agent_lease_extend frames.

ttl_ms (integer, REQUIRED, greater than 0, at most 3 600 000) The advisory lease lifetime in milliseconds.

intent (string, OPTIONAL, up to 2048 octets) A human-readable reason for the claim.

6.5. agent_lock_release

lease_id (string, REQUIRED) The lease the announcement releases.

resource (string, REQUIRED, 1 to 512 octets) The resource the released lease covered.

6.6. agent_lease_extend

lease_id (string, REQUIRED) The lease whose lifetime is extended.

additional_ttl_ms (integer, REQUIRED, greater than 0, at most

3 600 000) The additional lifetime in milliseconds.

6.7. agent_query

query_text (string, REQUIRED, 1 to 2048 octets) The question.

query_id (string, REQUIRED) A version-4 UUID correlating the query with its responses and return event.

response_scope (string, REQUIRED, 1 to 512 octets) The recipient scope (Section 6) to which responses are to be delivered.

timeout_ms (integer, REQUIRED, greater than 0) The caller-side window, in milliseconds, over which the caller collects responses.

6.8. agent_response

query_id (string, REQUIRED) The query this frame answers.

response_text (string, REQUIRED, 1 to 2048 octets) The answer.

responder (string, REQUIRED, 1 to 128 octets) An identifier for the responding session.

6.9. agent_return_event

return_event_ref (string, REQUIRED, 1 to 256 octets) A reference to a return record on the implementation’s return-accounting surface.

query_id (string, OPTIONAL) The query this return event closes, where the return is paired with an agent_query.

summary (string, REQUIRED, 1 to 2048 octets) A human-readable summary of what was returned.

6.10. agent_binding_moment

The agent_binding_moment payload carries an eight-slot decision request. The eight slots are: a one-sentence synopsis; an array of findings; an array of recommendations; a one-line offer; and a question object comprising a one-sentence stem, an array of two to four options (each an object carrying a short label and a one-line reasoning), a recommended_idx integer naming exactly one option, and a hatches object carrying two booleans, free_text and dialogue, each defaulting to true.

The two hatches preserve a dual escape from the decision frame: free_text indicates the recipient may answer outside the option set, and dialogue indicates the recipient may revise the question itself. A decision-request frame in which both hatches are false is malformed; a recipient MUST be left a path that is not one of the enumerated options.

6.11. peer_diagnostic_request

symptom (string, REQUIRED, 1 to 2048 octets) A description of the observed symptom.

diagnostic_id (string, REQUIRED) A version-4 UUID correlating the request with its response.

substrate_refs (array of strings, OPTIONAL) Observation references supporting the report.

severity (string, REQUIRED) One of "info", "degraded", "blocked".

6.12. peer_diagnostic_response

diagnostic_id (string, REQUIRED) The request this frame answers.

finding (string, REQUIRED, 1 to 2048 octets) The diagnostic finding.

remediation (string, OPTIONAL, up to 2048 octets) A suggested remediation, where one is known.

6.13. intent_declare, intent_withdraw, flush_executed

intent_declare carries a convergence_class namespaced identifier, a payload_ref pointer to the intended effect, an acted_by and drafted_with attribution pair, a declared_at timestamp, a ttl decay horizon in milliseconds, a withdrawable boolean, and an OPTIONAL urgency of "normal" or "urgent".

intent_withdraw carries the convergence_class, an intent_ref pointing at the declared intent, and a withdrawn_at timestamp.

flush_executed carries the convergence_class, a result_ref pointing at the result of the executed effect, an OPTIONAL batch_refs array of the included intent pointers, and an executed_at timestamp.

The pointer fields (payload_ref, intent_ref, result_ref, batch_refs) carry references (change-request identifiers, settlement identifiers, filing identifiers) and not the referenced objects themselves. Carrying references keeps frames small and keeps the delivery substrate free of the referenced content.

7. Recipient Scope Grammar

A frame is submitted with a recipient scope: a string naming the set of sessions the frame is to reach. The delivery substrate expands the scope at delivery time against the live subscriptions (Section 7) of recipient_handle. A scope is not a list of endpoints; the sender names a set by its identity shape, and the substrate resolves the membership.

The following scope forms are specified.

~handle Point-to-point in the degenerate sense: every session of the named handle. Equivalent to ~handle/*.

~handle/* All sessions of the named handle. When ~handle is the sender’s own handle, this is the same-principal fan-out that reaches all the sender’s concurrent sessions.

~handle/<prefix>* All sessions of the named handle whose instrument identifier begins with <prefix>. For example, ~alice/cc-* reaches all of ~alice’s sessions whose instrument identifier begins cc-.

~handle/<instrument>@<session-id> A single named session, identified by instrument and session identifier.

org:<org-handle>/members/* All sessions of all recognised members of the named organisational identity substrate.

org:<org-handle>/members/<role>/* All sessions of recognised members holding the named role.

accord:<peer-org-handle>/grant:<grant-scope> All sessions of the named peer organisation reachable under an Identity Accord [IDACCORD] grant of the named grant scope.

A delivery substrate MUST refuse a scope the submitting session is not authorised to address. In particular, a session MUST NOT submit a frame to a scope naming a handle other than recipient_handle, MUST NOT submit a frame to an org: scope of an organisation that does not recognise the sender as a member, and MUST NOT submit a frame to an accord: scope absent a corresponding Accord grant. Cross-organisational delivery is always grant-mediated through the peer-protocol ceremony of [IDACCORD]; it is never effected by an administrative assignment.

A delivery substrate MAY decline to implement the org: and accord: scope forms in an initial deployment, in which case it MUST reject a frame carrying an unimplemented scope with the scope-unimplemented error of Section 10 rather than silently dropping it.

8. Delivery over the Per-Handle Stream

Each ~handle is associated with one per-handle Server-Sent Events stream HTML-SSE [RFC8441]. A session of the handle opens the stream by an authenticated request to the delivery substrate resolved per [MCPDNS]. The stream is unidirectional from the substrate to the subscriber; a session sends a frame by a separate authenticated submission, not over the stream it subscribes on.

8.1. Submission and Expansion

A session submits a frame to the delivery substrate. The substrate:

1. validates the frame against Sections 3, 4, and 5, rejecting a malformed frame per Section 10;

2. verifies that sender_handle matches the submitting session’s authenticated identity, and that the submitting session is authorised to address the supplied recipient scope (Section 6);

3. expands the recipient scope against the live subscriptions of recipient_handle; and

4. emits the frame as a Server-Sent Events event on the per-handle stream of each subscription the expansion names, subject to that subscription’s filter (Section 8).

A frame whose expanded scope names no live subscription is delivered to no subscriber. This is not an error: the sender addressed a set, and the set was, at delivery time, empty. The submission result reports the count of subscriptions the frame was emitted to so a sender may observe an empty fan-out.

8.2. Stream Maintenance

The per-handle stream is a long-lived HTTP response. The delivery substrate SHOULD emit a periodic keepalive comment so that intermediaries do not close an idle stream, and a subscriber SHOULD treat a prolonged absence of both frames and keepalives as a dropped stream and re-open it.

Each emitted event carries the Server-Sent Events id field. A subscriber that re-opens a dropped stream MAY supply the id of the last event it received in the Last-Event-ID request header. A delivery substrate MAY honour Last-Event-ID by resuming emission after the named event, where the substrate retains recent events; a substrate that does not retain events, or for which the named event has aged past retention, resumes emission from the current stream position. Whether, and for how long, a substrate retains delivered frames for resumption is an implementation matter and is not specified by this memo (Section 1). A subscriber MUST NOT assume gap-free resumption.

8.3. Ephemerality

A delivered frame is a routing unit. Once the delivery substrate has emitted a frame to the subscriptions its scope named, this memo places no further obligation on the substrate with respect to that frame. The frame’s ttl_ms is advisory to recipients and is not a directive to the substrate. An implementation that retains delivered frames does so under its own retention policy; the retention policy, the order in which retained frames are presented, and any durability property of the retention store are outside the scope of this specification.

9. Subscribe Filter Grammar

A subscriber MAY narrow the frames it receives on its per-handle stream by supplying a filter expression when it opens the stream. The filter is carried as a query-string parameter on the stream-open request; this memo names the parameter filter for the reference deployment, and an implementation MAY use any name consistent with its addressing convention.

A filter expression is a comma-separated list of key:value clauses. A frame is emitted to a filtered subscription only if it satisfies every clause. An empty filter, or an absent filter parameter, matches every frame.

The following filter axes are specified.

kind:<kind> Matches frames whose kind equals the named catalogue value. A value outside the catalogue MUST cause the stream-open request to be rejected, so that a misspelled kind does not silently match nothing.

sender:<~handle> Matches frames whose sender_handle equals the named handle. A value that is not a canonical handle MUST cause the stream-open request to be rejected.

content_type:<value> Matches frames whose payload carries a content-type field equal to the named value. A frame whose payload carries no content-type field does not match a content_type clause; the axis is an opt-in field filter, not a default-pass.

tool:<tool-class> Matches frames emitted by a session of the named tool class.

org:<org-handle> Matches frames scoped to the named organisation.

tool and org are forward-compatibility axes: a deployment that has not yet implemented the org: recipient scope of Section 6 accepts and parses an org filter clause so that subscribers may write their filter strings ahead of substrate support, but the clause matches no frame until the corresponding scope form is implemented. A parsed-but-not-yet-effective clause MUST NOT widen the stream; the subscriber asked for a narrowing and receives, at worst, a valid no-op.

An unrecognised filter axis MUST cause the stream-open request to be rejected with the filter-axis-unknown error of Section 10. A filter is a narrowing instruction the subscriber relies on; a typo that silently disabled the narrowing would be a privacy failure, not a convenience.

Filtering is performed by the delivery substrate, not by the subscriber. A subscriber that supplies a filter receives only the frames the filter admits; the substrate does not emit and then expect the subscriber to discard. Substrate-side filtering keeps the volume on a constrained subscriber link proportional to what the subscriber asked for, rather than proportional to the handle’s total fan-out traffic.

10. Error Handling

A delivery substrate that rejects a submitted frame, or a stream-open request, MUST return a machine-readable error. An error is a JSON object carrying a stable code string, a field naming the offending field where one applies, and a human-readable message. The following codes are specified.

envelope-version-unsupported The frame’s envelope_version is not implemented.

kind-unknown The frame’s kind is outside the fifteen-value catalogue.

payload-kind-mismatch The frame’s payload shape does not match its kind.

field-missing A REQUIRED field is absent. field names it.

field-invalid A field carries a value outside the constraints of Section 3, 4, or 5. field names it.

field-unknown The frame carries a field outside the specified set (Section 12). field names it.

sender-identity-mismatch The frame’s sender_handle does not match the submitting session’s authenticated identity.

scope-unauthorised The submitting session is not authorised to address the supplied recipient scope.

scope-unimplemented The supplied recipient scope is well-formed but the delivery substrate does not implement that scope form.

filter-axis-unknown A stream-open filter clause names an unrecognised axis.

filter-value-invalid A stream-open filter clause carries a value invalid for its axis.

A rejection MUST NOT result in a partial delivery: either a frame is fully validated and entered into scope expansion, or it is rejected and emitted to no subscriber.

11. A Reference Tool Surface

This section describes, non-normatively, the agent-runtime-facing verb surface of the reference deployment. The verbs are exposed as Model Context Protocol [MCP] tools; an implementation MAY expose the same operations under any addressing convention. The verbs are named here so that the wire elements of Sections 3 through 8 can be related to the operations an agent runtime performs.

agent_send Submits a single frame to a recipient scope.

agent_advise Submits an agent_advisory frame; a distinct verb from agent_send so a runtime hook can emit an advisory without selecting a kind.

agent_broadcast Submits an agent_broadcast frame to a fan-out scope.

agent_handover Submits an agent_handover frame, retiring the temporary-file fallback of Section 1.

agent_lock_acquire, agent_lock_release, agent_lease_extend Submit the three advisory lock-shaped frames of Section 4.4.

agent_query Submits an agent_query frame and returns the query identifier for the caller to correlate responses against.

agent_roster Returns the set of the handle’s sessions currently subscribed to the per-handle stream, i.e. the live membership against which a recipient scope would expand.

agent_subscribe Opens the per-handle stream with an OPTIONAL filter expression (Section 8).

The reference deployment exposes these verbs alongside, and distinct from, the verb surface by which human-readable messages are exchanged. The agent-channel frame and the human-message surface share a delivery substrate and a per-handle stream; they are distinguished by the structured-versus-prose shape of their payloads and by the kind catalogue, which the human-message surface does not carry.

12. Composition with Companion Memos

This memo composes with four Morrison-family Internet-Drafts and does not introduce a transport, a handle category, or an attribution slot beyond them.

[IDPRONOUNS] supplies the ~handle namespace and the Sovereign/ Instrument trust-tier taxonomy used by the sender_handle, recipient_handle, acted_by, and drafted_with fields and by the scope grammar of Section 6.

[MCPDNS] supplies the discovery mechanism by which an agent runtime resolves the delivery substrate for a ~handle and opens the per-handle stream against it.

[IDCOMMITS] supplies the tier-structured attribution grammar that the acted_by / drafted_with pair of Section 3.2 carries at wire time. A frame's attribution pair and a commit's trailer block are the same shape applied at two points in a workflow.

[IDACCORD] supplies the peer-protocol ceremony that mediates the accord: recipient scope of Section 6. Cross-organisational delivery is grant-bounded through that ceremony; no meta-federation authority is introduced.

[SUBSTRATE] supplies the coordination posture this memo deliberately does not duplicate. The present memo is a delivery frame for messages a session chooses to send; it is not a coordination protocol. Where concurrent sessions of a principal must deconflict conflicting action, they do so through the substrate-observation cascade of [SUBSTRATE]. The lock-shaped kinds of Section 4.4 are announcements within the present memo's delivery model and are explicitly not the enforcement mechanism; [SUBSTRATE] specifies how conflicting action is reconciled.

13. Extensibility

The agent-channel frame object does not admit extension fields. An implementation that receives a frame carrying a field outside the set specified in Sections 3, 4, and 5 MUST reject it with the field-unknown error of Section 10. Closed-object validation is deliberate: a frame is a small routing unit on a constrained stream, and a silently-accepted unknown field is more likely a sender bug than a forward-compatible extension.

Extension of the protocol proceeds by revision of this memo. A new frame kind, a new payload field, or a new scope or filter form is introduced by a new envelope_version, and a receiver that does not implement the new version rejects frames carrying it (Section 3.1) rather than parsing them partially. The fifteen-value kind catalogue of Section 4 is, for envelope_version "1.0", exhaustive.

14. IANA Considerations

This memo requests no IANA action.

The frame format of this memo is an application-layer JSON object [RFC8259] carried over the existing Server-Sent Events transport HTML-SSE [RFC8441]; it introduces no new media type, no new URI scheme, no new port, and no new DNS resource record. The per-handle stream is discovered and addressed by the mechanism of [MCPDNS] and requires no separate allocation here.

The fifteen kind values of Section 4, the recipient-scope forms of Section 6, the filter axes of Section 8, and the error codes of Section 10 are protocol constants of this specification. This memo does not request an IANA registry for them. Should a future revision, or a companion specification, propose a registry for the kind catalogue or the error codes (for example to permit third-party kind allocation), that revision will request the corresponding IANA action and specify the registration policy.

15. Security Considerations

15.1. Sender Spoofing

A frame asserts a sender_handle and an acted_by / drafted_with attribution pair. An unauthenticated submission path would let any party assert any handle. The delivery substrate MUST authenticate the submitting session and MUST verify that sender_handle matches the authenticated identity before the frame enters scope expansion; a frame failing the check is rejected with sender-identity-mismatch (Section 10). The substrate SHOULD additionally verify that the acted_by handle is the Sovereign-tier handle the authenticated session is bound to, so that an Instrument-tier session cannot assert a Sovereign-tier acted_by other than its own principal’s.

15.2. Scope-Authorisation Bypass

The recipient-scope grammar of Section 6 lets a sender address a set rather than an endpoint. A sender that could address an arbitrary scope could deliver unsolicited frames to any handle or any organisation’s members. The delivery substrate MUST refuse a scope the submitting session is not authorised to address: a scope naming a foreign handle, an org: scope of an organisation that does not recognise the sender, or an accord: scope absent a grant. Authorisation is checked at submission time, before scope expansion; a frame failing the check is rejected with scope-unauthorised and reaches no subscriber.

15.3. Fan-Out Amplification

A single submitted frame to a broad scope (org:<org-handle>/members/*) expands to many emitted events. An attacker in possession of a member credential could submit frames at a high rate to amplify load across an organisation's subscribers. Delivery substrates SHOULD rate-limit submissions per authenticated sender, SHOULD cap the cardinality a single scope expansion may produce, and SHOULD apply the per-frame size bounds of Section 5 so that an amplified frame is at least small. The convergence kinds of Section 4.7 are themselves a load-reduction mechanism (they let a set of sessions collapse many intended effects into one), and a substrate MAY treat a high rate of intent_declare frames without corresponding flush_executed frames as an abuse signal.

15.4. Lock-Shaped Frames Are Not Authority

The agent_lock_request, agent_lock_release, and agent_lease_extend kinds are advisory announcements (Section 4.4). A reader, or an implementer, who treats them as an enforcement mechanism introduces a security defect: a recipient that _relies_ on a received agent_lock_request to guarantee exclusive access has trusted an unenforceable announcement, and an attacker who suppresses or forges such a frame can induce a recipient to act, or to refrain from acting, incorrectly. Mutual exclusion, where an implementation requires it, MUST be enforced by a local mechanism (a filesystem advisory lock, an operating-system lease) that does not depend on frame delivery. The lease identifier carried by these frames is a correlation value and MUST NOT be treated by the delivery substrate as a key into a lock table.

15.5. Replay and Duplication

A frame carries a sender-generated frame_id. An attacker positioned on the transport could re-deliver an aged frame to refresh a recipient's view of stale state, for example by replaying an agent_advisory to make a long-departed session appear active. The per-frame ttl_ms bounds the window in which a recipient treats a frame's content as operative, and a recipient SHOULD discard a frame whose created_at plus ttl_ms lies in the past. A recipient SHOULD additionally de-duplicate by frame_id so a duplicated delivery of the same frame is processed once. Because the lock-shaped frames are not authority (above), a replayed lock-shaped frame cannot, by itself, grant or revoke access; its worst effect is a stale advisory, bounded by ttl_ms.

15.6. Diagnostic-Frame Information Disclosure

A peer_diagnostic_request carries a symptom description and observation references; a peer_diagnostic_response carries a finding and a remediation. These may disclose configuration, file paths, or operational detail a sender did not intend to expose beyond its own sessions. A sender SHOULD scope diagnostic frames no more broadly than the troubleshooting requires, preferring point-to-point or its own sessions over an org: scope, and SHOULD redact secret material from a symptom description before submission.

16. Privacy Considerations

16.1. Recipient-Scope Membership Inference

The submission result of Section 7.1 reports the count of subscriptions a frame was emitted to. A sender that submits frames to successively narrower scopes can, from the reported counts, infer how many of a handle’s sessions, or an organisation’s members’ sessions, are currently online. A delivery substrate SHOULD consider whether to report an exact count or a coarsened one, and SHOULD restrict the agent_roster operation of Section 9, which enumerates a handle’s live sessions, to the handle’s own sessions and to parties the handle has authorised.

16.2. Attribution Exposure

Every frame carries an acted_by Sovereign-tier handle and a drafted_with Instrument-tier handle. Within a fan-out, every recipient of a frame learns the sending principal’s identity and the runtime that composed the frame. This is intentional; the attribution pair is the basis on which a recipient assesses a frame. A sender should be aware that an org: or accord: scope discloses the attribution pair to every member of the addressed set. A sender that wishes to limit attribution exposure SHOULD prefer the narrowest scope that reaches the intended recipients.

16.3. Provenance-Block Content

The provenance block of Section 3.3 carries a compute-location, a set of derivation-method tags, and a stream-basis tag. These describe how the frame’s content was produced and may, in aggregate across many frames, characterise a sender’s working pattern. The block is carried for the recipient’s benefit; a sender SHOULD populate provenance_method and provenance_basis with tags no more specific than a recipient needs to assess the frame, and an implementation SHOULD NOT require the block to carry identifying detail beyond the derivation method.

16.4. Ephemerality as a Privacy Property

This memo specifies delivery, not storage (Section 1, Section 7.3). A frame delivered and not retained leaves no standing record on the delivery substrate beyond the substrate's transient processing of it. Where an implementation chooses to retain delivered frames for stream resumption, that retention is a privacy-relevant decision the implementation makes outside this specification: a retained frame is a record of who said what to whom, and the retention horizon, access controls, and deletion behaviour of any such store SHOULD be specified by the implementation and surfaced to the principals whose frames it retains. The convergence kinds of Section 4.7 carry an explicit decay horizon (ttl on intent_declare) precisely so that the announcement of an intended effect is, by design, forgotten once the window for the effect has closed.

17. References

17.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8441] McManus, P., "Bootstrapping WebSockets with HTTP/2", RFC 8441, DOI 10.17487/RFC8441, September 2018, <https://www.rfc-editor.org/info/rfc8441>.

[MCPDNS] Morrison, B., "Discovery of Model Context Protocol Servers via DNS TXT Records", 2026, <https://datatracker.ietf.org/doc/draft-morrison-mcp-dns-discovery/>.

[IDPRONOUNS] Morrison, B., "Identity Pronouns: A Reference-Axis Extension to ~handle Identity Systems", 2026, <https://datatracker.ietf.org/doc/draft-morrison-identity-pronouns/>.

[IDACCORD] Morrison, B., "Identity Accord Protocol", 2026, <https://datatracker.ietf.org/doc/draft-morrison-identity-accord/>.

[IDCOMMITS] Morrison, B., "Identity-Attributed Git Commits via Tier-Structured Trailers", 2026, <https://datatracker.ietf.org/doc/draft-morrison-identity-attributed-commits/>.

[SUBSTRATE] Morrison, B., "Substrate-Observation as an Alternative to Envelope Coordination for Concurrent Sessions", 2026, <https://datatracker.ietf.org/doc/draft-morrison-substrate-observation/>.

[MCP] Agentic AI Foundation, "Model Context Protocol Specification", 2026, <https://modelcontextprotocol.io>.

17.2. Informative References

[HTML-SSE] WHATWG, "HTML Living Standard, Section 9.2: Server-Sent Events", 2026, <https://html.spec.whatwg.org/multipage/server-sent-events.html>.

Acknowledgements

This memo grew out of internal architectural work on how the several concurrent agentic sessions of a single principal, today reduced to passing a file through a shared temporary directory, should instead exchange short structured messages over an identity-keyed delivery substrate. The realisation that the needed primitive is a fan-out _delivery_ frame, and that coordination, as distinct from delivery, is better left to substrate observation than standardised as an envelope, is the load-bearing insight behind this specification and its companion memo [SUBSTRATE].

Author’s Address

Blake Morrison
Alter Meridian Pty Ltd (~truealter)
Email: blake@truealter.com
URI: alter:~blake