

The Service Handshake Technical Specification

An open interaction standard for service interactions where one or more parties are represented by AI agents

Maria McCann, Founder, Neos Wave

SH-1.2 | Published 1 October 2026

Version 1.1: [10.5281/zenodo.19046746](https://zenodo.org/record/19046746).

Version 1.2: [10.5281/zenodo.23181969](https://zenodo.org/record/23181969).

Licence: Creative Commons Attribution 4.0 International (CC BY 4.0). Schemas, examples and tests: Apache License 2.0. © 2026 Neos Wave Ltd. Contact: neoswave.com.

About this revision

This is the complete revised specification, retaining the six elements, field references and worked examples from SH-1.1. Minor editorial amendments improve clarity and consistency without changing requirements and are not listed individually.

Key changes from SH-1.1

- **Participation and incomplete declarations.** Assisted participation replaces automatic rejection. Unknowns restrict the affected action; assistance and a human route remain available. Two conformance profiles support different implementation needs. See §§1.2 and 2.4.
- **Disclosure and provenance.** New metadata records AI disclosure, preparation, confirmation, revisions and unresolved fields. Confirmation establishes a claim without granting authority. See §2.3.
- **Data permissions and confidence.** Service-purpose limits last throughout the interaction. Confidence scores are optional and cannot establish authority; the general 0.85 recommendation is withdrawn. See §5.
- **Safeguarding and handover.** Baseline indicators and relayed concerns support nonwaivable harm routing. Handovers and linked contacts preserve restrictions. The separate vulnerability module remains deferred. See §§6.3–6.5.
- **Authority and financial controls.** Configuration does not establish delegation. Approval requires independently checkable evidence appropriate to the action. Existing procedures can qualify; internal thresholds may remain private. See §8.
- **Resolution compatibility.** Assess the proposed resolution using common meanings and complementary roles, including cancellation and appointments. An unused refused option does not invalidate an acceptable alternative. See §9.3 and Appendix A.
- **Implementation and security.** General controls replace platform-specific recipes, including Make.com. Incoming declarations remain untrusted data and cannot set privileged instructions or permissions. See §§10 and 13.

- **Schema and example corrections.** Corrections cover fallback arrays, precedence, retention values, family roles and incompatible example terms. Empty lists and optional confidence scores support incomplete participation without granting permission. See §§2–9 and 12.
- **Version support and decision records.** Each version uses its own schema. Separate assessments record decisions and execution. Purpose-based retention replaces the blanket twelve-month rule for change records. See §§11–12 and 14.

1 Purpose and scope

This specification defines the Service Handshake Declaration: the record of terms established for a service interaction where at least one party is represented by an AI agent. The main paper explains the design philosophy, four modes and six elements. This companion specifies their implementation for developers, architects and platform teams.

It covers field definitions, machine validation, assisted participation, retail and triadic examples, compatibility assessment, workflow controls and version handling. JSON is the machine-exchange format. Staff-operated Participation can use an equivalent case form.

The uppercase requirement words **MUST**, **MUST NOT**, **REQUIRED**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY** and **OPTIONAL** use the meanings in BCP 14, RFC 2119 and RFC 8174. Lowercase uses have their ordinary meaning. [1, 2]

SH does not issue credentials, provide transport, determine the legal validity of a delegation or execute transactions. An implementation claiming conformance **MUST** connect the required checks to the people and systems controlling the protected action. Displaying a declaration or generating an advisory score is insufficient.

1.1 Terms

The principal is the person or organisation represented for the relevant act. A beneficiary is a person or organisation affected by or receiving the result. A representative acts for a principal. The agent's configurator need not hold authority over the principal's affairs. The relying party is the party deciding whether to accept evidence or permit an action.

A claim is an assertion made by a party. Evidence is material supporting a claim. A decision is the relying party's recorded assessment of a particular action. An outcome is what actually happened. These categories **MUST NOT** be collapsed into a single verified flag supplied by the requesting party.

1.2 Meeting the standard

The Service Handshake has two levels of requirements:

Participation covers the essential safeguards: disclosing AI involvement at first contact, offering assistance where a declaration is missing, establishing and respecting both parties' boundaries, checking authority, limiting data use to the service purpose, responding to safeguarding concerns and providing a human route that preserves restrictions. Missing information **MUST NOT** be treated as permission. These requirements can be supported through existing staff procedures and case records.

Verified authority includes all Participation requirements and adds independently checkable evidence of authority, linked to the specific action. It also requires checks that the evidence remains valid and has not been withdrawn, controls to prevent an approval being reused improperly, and a record of the action assessment.

Organisations **MUST** state which level they meet and which services and actions their claim covers. A Verified authority claim **MUST** also identify the evidence mechanisms and execution paths covered. It does not cover every action handled by an organisation.

Participation still requires authority checks. Existing trusted staff or account procedures **MAY** provide the required evidence, but an implementation **MUST NOT** claim Verified authority unless it meets the additional requirements. Where required evidence is unavailable, the affected action **MUST** be held or referred to a process capable of completing the checks.

2 Schema overview

A declaration is a JSON object. The minimum structural fields remain `protocol_version`, `principal_id`, `agent_id` and `goals`. Structural validity does not establish that sufficient information exists for any particular action. Each relevant boundary, permission and authority **MUST** be established before protected access or execution. Machine exchange is primarily used in Modes 3 and 4. Disclosure and company-agent controls also apply in Mode 2, and restrictions survive handover into Mode 1. A change of mode **MUST NOT** bypass applicable requirements.

2.1 Top level structure

The top level contains the identity and version fields, the six declaration objects, and the SH-1.2 metadata listed below. The examples in §9 show complete objects. Optional objects are omitted when unknown, with unresolved fields recorded; required identity or goal information that has not yet been established belongs in an intake record rather than a fabricated declaration.

2.2 Top level field reference

Field	Type	Status	Meaning
protocol_version	string	Required	SH-1.2 for this representation.
principal_id	string	Required	Reference to the person or organisation represented for this action.
agent_id	string	Required	Agent identifier, or a case-scoped representative reference for a human record.
beneficiary_id	string	Optional	Person or organisation receiving or affected by the outcome; may match the principal.
identity_tokens	array	Optional	Issuer and token-type references with asserted scope, status or expiry. Incoming values require trusted verification.
issuer	string	Optional	Who issued the declaration; does not itself establish the issuer of authority evidence.
valid_from / valid_to	date-time strings	Optional	Declaration validity window. Check ordering and applicable expiry.
Six elements	objects	Goals required	goals, options_constraints, data_permissions, fallback_rules, cost_parameters and authority_level.

`protocol_version` is SH-1.2 for this version. `principal_id` identifies the represented party and `agent_id` identifies the agent or the human representative's case reference. A scoped reference may be used during initial participation; it is not proof of identity. Implementations MUST NOT invent a verified identity to satisfy a schema requirement. If the represented party or goal is unknown, keep an intake record until those required fields can be supplied.

`beneficiary_id`, `issuer`, `identity_tokens`, `valid_from` and `valid_to` retain their SH-1.1 types. Incoming identity-token status values remain claims. Their truth MUST be established by a trusted verifier when needed. The relying party MUST check time ordering and relevant expiry, not merely date formatting. Duplicate principal or beneficiary identifiers inside `authority_level` MUST match the canonical top-level values.

2.3 Preparation and confirmation metadata

Field	Representation	Meaning
declaration_id	String	Reference to the declaration; pair with its revision for action binding.
declaration_revision	Positive integer	Increment when material terms change.
declaration_provenance	self_declared, pre_agreed or assisted	How the declaration was established; not its assurance level.
disclosure	Object	AI status, representative role, disclosure time and method when known.
confirmation	Object	Confirmation status, confirmer reference, time and method when available.
unresolved_fields	Array of JSON Pointer strings	Unknown or ambiguous fields that cannot be treated as permission.

`self_declared` means terms issued by the declaring party without bilateral prior agreement or assistance. `pre_agreed` means terms previously agreed between the relevant parties for the stated scope; it is not restricted to B2B. A standing company policy is self-declared unless that prior agreement exists. `assisted` means terms prepared with the other party's help and presented for confirmation. None of these values denotes verified authority.

Provenance SHOULD be recorded for all declarations and MUST be recorded as `assisted` for assisted declarations. Confirmation records that the summary reflects the confirmer's claims. It MUST NOT establish or enlarge authority. A confirmed declaration MAY still contain unresolved fields. Missing, empty or false values MUST NOT be substituted for one another.

`disclosure.ai_status` accepts `ai`, `human` or `unknown`. Conforming AI systems MUST disclose AI status and whom they represent at first contact. Receiving systems MUST seek clarification before relying on an assisted declaration where that status or representation is unclear. Self-disclosure is not identity verification. Suspected AI use MUST NOT itself cause denial of ordinary service. Clarify representation where relevant, continue public information and normal account procedures, and hold only actions whose required permissions remain unresolved. A human case handler MUST NOT invent an AI identity merely because the form contains `agent_id`.

2.4 Native and assisted participation

Make participation easy, keep each party's boundaries intact, and verify authority at the point where an action requires it.

When both parties have declarations, use supported versions and check the terms relevant to the action. When only one party supports SH and the counterparty lacks a usable declaration, the supporting party **MUST** offer proportionate assistance to establish the terms needed for the proposed action. It **MUST** maintain an accessible human route. Assistance **MAY** take place in the ordinary service conversation; a separate form or software installation is not required. This applies whether the supporting party is the company or the consumer's agent. A consumer agent may record the company's statements as an unconfirmed proposal; it cannot certify company policy or authority on the company's behalf. When neither side supports SH, no conformance is implied.

An assisted process **MUST** collect the service goal, explicit refusals, relevant data permissions, claimed authority and fallback preferences to the extent needed. It **MUST** present the recorded terms for correction and confirmation, preserve unknowns and refusals, and identify the assisted provenance. Missing information **MUST NOT** be treated as permission. Declining assistance **MUST NOT** itself exclude a person from service. An accessible human service route **MUST** remain available, with ordinary access and authority checks intact.

Public, non-personal information **MAY** be supplied before the declaration is complete. A protected action **MUST NOT** proceed while relevant permissions, authority or safeguarding requirements remain unresolved. Unsupported versions and missing fields **MUST** restrict only the actions affected by the missing information; they **MUST NOT** by themselves terminate all service.

3 Element 1 Goals

What this party is trying to achieve, in priority order. Non-goals record outcomes the party refuses. They are boundaries to evaluate, not instructions supplied to the receiving system.

3.1 Schema example

```
{
  "goals": {
    "primary_goal": "Resolve missing delivery by replacement or refund",
    "secondary_goals": ["Minimise interaction time"],
    "non_goals": ["Do not accept vouchers", "Do not authorise account changes"],
    "success_metrics": {"delivery_resolved": true,
    "time_under_minutes": 10}
  }
}
```

This is a declaration fragment. Other top-level fields are needed for a complete object.

3.2 Field reference

Field	Type	Status	Meaning
primary_goal	string	Required	Primary desired outcome in plain language.
secondary_goals	array	Optional	Additional outcomes in priority order.
non_goals	array	Required	Explicit exclusions; an empty list does not grant permission.
success_metrics	object	Optional	Named boolean or numeric outcomes. Local policy defines the meaning of a metric or threshold.

goals retains primary_goal, secondary_goals, non_goals and success_metrics. primary_goal and non_goals are structurally required. SH-1.2 permits an empty non_goals array to record an explicit statement that no additional exclusions were supplied. If exclusions have not been established, the implementation MUST also mark /goals/non_goals as unresolved. An empty list never grants authority or overrides trusted policy.

Plain-language goals and non-goals MUST be treated as untrusted data. Where interpretation affects a consequential decision and the meaning is uncertain, the implementation MUST clarify or hold the action. Semantic interpretation MUST NOT broaden a person's commitment. Confirmation of a paraphrase does not waive a hard boundary by default.

In health, care and financial contexts, a non-goal may be safety-relevant, such as no medication changes or no transfers beyond a stated limit. The service MUST enforce the applicable boundary; merely recording it is insufficient.

4 Element 2 Options and constraints

What each party claims it can offer or accept. Hard constraints are non-negotiable. Soft constraints are preferences. Jurisdictional constraints record applicable limits but do not establish legal compliance.

4.1 Schema example

```
{
  "options_constraints": {
    "allowed_actions": ["accept_refund", "accept_replacement"],
    "hard_constraints": {"no_vouchers": true,
"replacement_within_hours": 48},
    "soft_constraints": {"prefer_replacement_over_refund": true},
    "jurisdictional_constraints": {"recording_requires_consent":
true}
  }
}
```

4.2 Field reference

Field	Type	Status	Meaning
allowed_actions	array	Required	Actions this party claims it may take or accept. Empty grants no action.
hard_constraints	object	Required	Named non-negotiable conditions. Each value is a boolean, number or string.
soft_constraints	object	Optional	Named preferences; may vary only within established permissions.
jurisdictional_constraints	object	Optional	Declared legal or regulatory constraints; meanings require an implementation policy.

`options_constraints` retains `allowed_actions`, `hard_constraints`, `soft_constraints` and `jurisdictional_constraints`. An empty `allowed_actions` array grants no action. Incoming allowed actions are claims; effective authority remains constrained by the relying party's trusted policy and evidence.

Implementations MUST use explicit mappings between complementary roles. `offer_refund` and `accept_refund` describe different sides of a compatible refund; literal string intersection is insufficient. Unknown action identifiers or hard-constraint meanings MUST NOT be ignored when deciding a protected action. Clarify, hold or escalate instead. Appendix A defines core action meanings and complementary roles. Implementations exchanging supported core actions MUST use those meanings or an explicit, documented mapping. They MUST communicate which actions they support; conformance does not require offering every core action. Local extensions MUST identify their namespace and meaning. Unknown meanings require clarification, without assuming permission or terminating all service.

Hard constraints MUST NOT be overridden by the counterparty. An authorised change by the party that set a constraint requires a new declaration revision and reassessment; approval of an action alone does not amend the constraint. Preferences may vary only within established permissions. A conflict affecting one candidate does not invalidate other candidates that satisfy both parties' applicable boundaries.

5 Element 3 Data permissions

What data may be used, for which purposes, with what retention and verification requirements.

5.1 Schema example

```
{
  "data_permissions": {
    "data_sources_allowed": ["order_reference"],
    "data_uses_allowed": ["problem_resolution"],
    "retention_policy": {"policy": "no_retention_beyond_case"},
    "verification_requirements":
["verify_access_before_order_lookup"]
  }
}
```

5.2 Field reference

Field	Type	Status	Meaning
data_sources_allowed	array	Required	Permitted categories of data. Empty grants no access.
data_uses_allowed	array	Required	Permitted service purposes. Other processing requires a separate applicable basis.
retention_policy	object	Required	retain_for_days, a supported policy name, or both where consistent.
trust_levels	object	Optional	Legacy confidence settings; self_trust is required inside this object if it is supplied.
verification_requirements	array	Optional	Additional verification steps needed before relevant access or action.

`data_permissions` retains `data_sources_allowed`, `data_uses_allowed`, `retention_policy`, `trust_levels` and `verification_requirements`. The first three are required when the object is present. Empty permitted-source or permitted-use arrays grant no access or purpose. An absent object leaves relevant permissions unresolved.

For the whole interaction, each party **MUST** limit handling under the handshake to the declared service purpose. A declaration **MUST NOT** itself authorise marketing, unrelated profiling, model training or unrelated onward sharing. Necessary security, fraud or statutory-retention uses **MUST** be identified separately with their basis and limits. Separately authorised processing outside the handshake cannot be inferred from participation or confirmation.

The supported retention policy names remain `no_retention`, `no_retention_beyond_case` and `retain_per_legal_requirement`. `retain_for_days` remains a non-negative integer. If named and numeric terms conflict, the relying party **MUST** resolve the conflict before relying on them. A declared retention term cannot override a legal obligation; the service **MUST** explain the applicable constraint and minimise retained material.

`trust_levels` is optional in SH-1.2. If supplied, `self_trust` retains its numeric representation for compatibility, but it is only an implementation-specific operational signal. Neither a high confidence score nor a counterparty trust claim may substitute for authority, access permission, evidence verification or safeguarding. The former universal recommendation of 0.85 is withdrawn.

Within `trust_levels`, `self_trust` is a number from 0 to 1; `counterparty_trust` is an optional object of boolean, numeric or string signals. The implementation defines their interpretation. A financial value in a trust object cannot replace the currency, scope, period and evidence needed for an actual commitment.

6 Element 4 Fallback rules

Fallback rules define what happens when the interaction cannot proceed within its terms. Implementations need uncertainty, conflict and harm routes; resource limits also need a next step.

6.1 Schema example

```
{
  "fallback_rules": {
    "fallback_triggers": [
      {"if_confidence_below": 0.7},
      {"if_unresolved_after_loops": 3},
      {"if_goals_conflict_on": "remedy"},
      {"if_counterparty_exceeds_scope": true},
      {"if_vulnerability_signal_detected": true}
    ],
    "fallback_actions": ["clarify_intent",
      "request_human_review"],
    "maximum_automation_scope": "No account changes without
      verified approval.",
    "responsibility_assignment": "Customer Care Team Lead",
    "precedence_logic": "CONSUMER_PRIORITY"
  }
}
```

The confidence value is illustrative, not a recommended universal threshold.

6.2 Field reference

Field	Type	Status	Meaning
fallback_triggers	array	Required	Array of single-condition objects; a harm-indicator trigger is required.
fallback_actions	array	Required	Ordered actions to attempt when routing is triggered.
maximum_automation_scope	string	Required	Boundary on actions the agent may initiate or complete without further approval.
responsibility_assignment	string	Required	Accountable role, resolved to a person by the case system.
precedence_logic	string	Optional	Escalation preference: CONSUMER_PRIORITY, BRAND_PRIORITY or PRE_NEGOTIATED_TREATY.

When supplied, `fallback_rules` requires an array of `fallback_triggers`, an ordered array of `fallback_actions`, `maximum_automation_scope` and `responsibility_assignment`. The trigger array MUST contain `if_vulnerability_signal_detected: true`; the schema enforces this presence. The relying party MUST also apply its own relevant uncertainty, conflict, cost and harm routes even when the incoming declaration omits the object.

`responsibility_assignment` names an accountable role. A separate case system may resolve that role to a person. `precedence_logic` is retained as an optional declared preference with values `CONSUMER_PRIORITY`, `BRAND_PRIORITY` and `PRE_NEGOTIATED_TREATY`. None overrides another party's authority, safeguarding or duties. The parties SHOULD select an accessible escalation owner before repeated exchanges occur; if they cannot agree, each remains responsible for its own safe next step and an explanation to the person.

Permitted trigger keys are `if_confidence_below` (number from 0 to 1), `if_unresolved_after_loops` (integer of at least 1), `if_goals_conflict_on` (string), `if_counterparty_exceeds_scope` (boolean), `if_vulnerability_signal_detected` (true), `if_contract_cancellation_proposed` (boolean) and `if_provider_switch_proposed` (boolean). Each trigger object contains one key. Local handling MUST define the meaning of each selected trigger.

For simultaneous escalation, the service SHOULD accommodate the consumer's requested route where compatible with the required safeguards and available authority. A declared precedence preference cannot make the other party's staff responsible without agreement. Both parties retain responsibility for their own safe next step.

6.3 Nonwaivable harm routing

An assisted declaration or counterparty approval **MUST NOT** disable the receiving party's applicable harm indicators. The service **MUST** define observable indicators, proportionate routing and an accountable human role for its context. The baseline below provides a starting point; services **SHOULD** obtain specialist input where risk or sector requirements call for it. Record the trigger and action taken, with minimum necessary sensitive detail. A flag **MUST NOT** become a speculative diagnosis or an unexplained restriction on service. Consequential restrictions based on a harm concern require an appropriate human review route.

The following baseline concerns call for review of the affected action. The service **MUST** provide a way for staff, a person using the service or an agent relaying that person's concern to raise them; automated inference is not required. A conforming consumer agent **SHOULD** relay these concerns when its principal expresses them. A company agent **MUST** treat a relayed concern as raised and apply the relevant safeguarding response. The record **MUST** distinguish a relayed report from an independently established fact, and disclose only the information needed for the response. A report does not itself establish a diagnosis or justify an indefinite restriction.

Observable concern	Baseline response
The person says they are being pressured, coerced or represented against their wishes.	Pause the affected commitment and provide a safe human route appropriate to the stated concern.
The person says they do not understand the terms or cannot give informed instructions in the current channel.	Explain or adapt the interaction, offer help and establish understanding before seeking approval.
The requested action may interrupt an essential service or the person reports an immediate safety concern connected to the service.	Refer to the designated urgent or specialist route; preserve safe access and permitted assistance.
The person's instructions conflict with the representative's request, or the representative's authority is disputed.	Hold the affected action and resolve the instructions and authority through the appropriate human process.

These indicators trigger assistance and review, not a diagnosis or a finding that a person lacks capacity. Age, disability, communication style or use of an agent **MUST NOT** alone justify restricting service. Additional sector duties and existing emergency procedures remain applicable. A service **MUST** record the concern, affected action, response and review route with minimum necessary sensitive detail.

The required `if_vulnerability_signal_detected: true` entry configures a fallback condition. It does not assert that a person is vulnerable or prove that detection works. The implementation **MUST** connect that condition to its documented indicators and test the routing; schema acceptance alone cannot satisfy this requirement.

This baseline does not deliver a sector-specific vulnerability taxonomy or diagnostic module.

The separate vulnerability module announced for SH-1.2 in SH-1.1 remains deferred.

6.4 Human handover

A handover MUST make visible the case reference, represented and affected parties as known, the action under consideration, applicable declaration references, each blocked action, its reason, evidence or clarification needed, data-use restrictions and the responsible role. Necessary safeguarding instructions MUST survive transfer with proportionate disclosure.

Transfer to a human MUST NOT itself remove a restriction. Any later permission to proceed MUST record the new evidence or authorised decision. A staff override MUST NOT create delegation that staff do not hold or waive an applicable nonwaivable requirement.

6.5 Restrictions across contacts

A fresh contact, channel, agent session or assisted declaration MUST NOT erase an unresolved restriction on the same case or proposed action. Where a case or account is reliably linked, the receiving process MUST check applicable open restrictions before protected access or execution. An unverified identifier MUST NOT expose another person's case or transfer a restriction to them.

The service MUST record each restriction's case or action scope, reason, owner, conditions for resolution and review point. Retention follows the service purpose and applicable obligations. A restriction MUST NOT become an unexplained, indefinite account-wide block; unaffected service and a review route remain available. Reaching a review date does not supply missing authority. If a channel cannot establish the applicable status, it MUST route the affected protected action to a process that can. Closing a hold requires the new evidence or authorised decision to be recorded.

7 Element 5 Cost parameters

What the interaction may consume in time, tokens or effort. These are resource budgets and MUST NOT be interpreted as permission to incur financial commitments.

7.1 Schema example

```
{
  "cost_parameters": {
    "max_duration_minutes": 10,
    "max_tokens": 500,
    "max_loops": 3,
    "escalation_on_cost": {"if_duration_exceeds_minutes": 8,
  "action": "offer_scheduled_callback"}
  }
}
```

7.2 Field reference

Field	Type	Status	Meaning
max_duration_minutes	number	Optional	Non-negative interaction duration. Recommended where a time limit applies.
max_tokens	integer	Optional	Non-negative integer token budget.
max_loops	integer	Optional	Non-negative integer exchange budget. Recommended to limit repeated negotiation.
escalation_on_cost	object	Optional	Object containing an action string and optionally if_duration_exceeds_minutes.

Approaching the applicable limit SHOULD trigger a summary, callback or other useful next step. If multiple limits apply, use the earliest applicable safe route. Financial exposure is addressed in §8.6.

8 Element 6 Authority level

Who the agent represents, who is affected, what it claims it can commit to and the evidence supporting that claim. The person who configured the agent is not automatically the principal or an authorised approver.

8.1 Schema example

```
{
  "authority_level": {
    "principal_id": "parent-account-holder-22",
    "beneficiary_id": "parent-account-holder-22",
    "authority_scope": ["com.example.utility:discuss_billing"],
    "authority_limits": ["no_cancellation",
"no_provider_switch"],
    "delegation_proof": "case-evidence-22-pending-verification",
    "interaction_role": "third_party_representative"
  }
}
```

This illustrative reference is not verified evidence.

8.2 Field reference

Field	Type	Status	Meaning
principal_id	string	Optional	Represented party for the action; if repeated here, MUST match the top-level value.
beneficiary_id	string	Optional	Affected or receiving party; if repeated here, MUST match the top-level value.
authority_scope	array	Required	Claimed scope of actions. An empty array grants nothing.
authority_limits	array	Required	Express limits on the claimed authority. Empty does not mean unlimited.
delegation_proof	string	Optional	String reference to evidence; its presence is not verification.
interaction_role	string	Optional	account_holder, carer, advocate, power_of_attorney, brand_agent or third_party_representative.
financial_limits	array	Optional	Applicable monetary limits, including currency, basis and policy reference; see §8.6.

`authority_level` retains its SH-1.1 fields. `authority_scope` and `authority_limits` remain arrays of claims; empty scope grants nothing. `delegation_proof` remains a string reference for compatibility. A phrase such as `account_holder_verified_via_2FA` or `power_of_attorney` MUST NOT be accepted as verified delegation solely because it appears in that field.

8.3 Proportionate requirements

Public information needs no personal account authority. Protected information requires relevant identity, access authority and data permission. Service actions require compatible terms and authority for the exact act. Additional controls follow from financial exposure, irreversibility, account security, essential-service consequences, harm risk and effects on other parties. Reversibility is not an exemption from authority checks.

Irreversible actions MUST have independently verifiable delegation or principal approval. A relying party MUST be able to validate that evidence without relying solely on the requesting agent's assertion. The standard does not require a separate app or device; a trusted approval surface within the same app can qualify if the evidence can be independently checked. The channel and evidence mechanism MUST be appropriate to the risk.

An existing staff or account procedure MAY supply this evidence under Participation. It MUST establish the approver's relevant authority, present the proposed action and material terms, obtain explicit approval, and record a result the relying party can validate independently of the

requesting agent. Cryptographic credentials are one possible mechanism; they are not the only route.

For example, a retailer may show the verified account holder the refund amount, original payment destination and any settlement conditions in its own authenticated approval process. The retailer records that person's approval against the proposal before staff issue the refund within company authority. A one-time code can be part of that process, but a code or answers to security questions alone do not demonstrate approval of those terms. A code read or supplied by the requesting agent **MUST NOT** be accepted as independent principal approval. The service **MUST** establish that the approval mechanism prevents that agent from approving on the person's behalf unless separate, sufficient delegation covers it.

This can complete an action through Participation. A claim of Verified authority additionally requires the controls and records in §§8.4–8.5 and 12.

8.4 Verification procedure

For Verified authority, the relying party **MUST** establish a trusted issuer or trusted approval source; verify authenticity using the applicable mechanism; identify the principal and requesting agent or authenticated session; check audience, resource, action scope and material parameters; check issuance, expiry and current status; and apply the mechanism's replay and revocation protections. A signature alone proves neither a trusted issuer nor sufficient scope.

For each relevant side, bind the evidence to the exact proposal and the material declaration versions used. If a bound term changes, reassess and obtain fresh approval where the evidence does not cover the change. Evidence unavailable, expired, revoked, out of scope or unverifiable **MUST NOT** yield an allow decision. If the mechanism has no revocation service, use its defined status method and appropriate short validity; if that cannot meet the action's requirements, hold the action.

Verification results **MUST** originate from a trusted local procedure, not an incoming declaration. The separate assessment schema structures those results but cannot authenticate them. Submitted records claiming to be local decisions remain untrusted until their source and integrity are established.

8.5 Execution boundary

The executor **MUST** check that an allow decision is applicable, current and bound to the operation being performed. It **MUST** also recheck dynamic conditions such as evidence status and financial usage before commitment where required. Use atomic reservation or equivalent concurrency controls for spending limits and one-time approvals. A stale decision or a retry **MUST NOT** create a second commitment. An uncertain execution result requires reconciliation before retrying.

8.6 Financial exposure

The optional `financial_limits` array belongs to `authority_level`. Each limit identifies an action type, maximum amount in integer minor units, currency, currency exponent and basis. The basis is `per_action`, `aggregate_period` or `recurring_total`. An aggregate-period limit requires its period in seconds. Policy defines the subject, resource scope, rolling or fixed window and calculation rules in a referenced policy version. If those semantics or the required ledger are unavailable, the affected financial action **MUST** be held.

For recurring proposals, record the interval and maximum number of occurrences. Unbounded recurring commitments are not represented by the action-assessment schema and require a separately specified review process. Zero means zero, never unlimited. Both parties' applicable limits **MUST** be satisfied. Currency conversion **MUST NOT** be inferred; a trusted, explicit conversion policy and its material terms are required. These fields are needed only where those financial controls apply.

8.7 Private internal limits

Parties **MAY** keep internal approval thresholds, fraud rules and negotiation limits private. The relying party **MUST** apply its applicable private limits in its own assessment and retain an auditable reference to the policy version and result. Omission of a limit from an exchanged declaration does not mean unlimited authority. The optional `financial_limits` field can communicate limits a party chooses to disclose; it is not a requirement to publish every internal threshold.

The actual offer and material conditions **MUST** be communicated clearly enough for the other party to decide. The service **MUST** explain a hold or refusal and the next step to the extent it can without exposing protected controls. Private limits **MUST NOT** be used to conceal material settlement conditions or bypass applicable obligations. An internal approval ceiling is not a commitment to offer that amount.

9 Worked examples and compatibility

The examples below are complete declaration objects, not complete approvals. All identifiers and evidence references are illustrative. Their corresponding JSON files are supplied in examples/. Structural validity does not authenticate a claim or authorise an action. The action-assessment records in that directory illustrate holds, refusals and safeguarding routes.

9.1 Mode 3 retail delivery resolution

A consumer agent contacts a retailer about a missing delivery. Staff establish AI disclosure and help prepare the declaration. The agent confirms that the summary reflects its claims; delegation remains unresolved. Public policy information is available, but protected lookup and refund execution still need the relevant checks. The retailer's representative is a staff member, so its record discloses a human and uses a case-scoped reference. The record states its own boundaries and an illustrative financial limit it has chosen to disclose. It is self-declared; no prior bilateral agreement is assumed.

Consumer declaration, assisted-consumer.json:

```
{
  "protocol_version": "SH-1.2",
  "declaration_id": "consumer-case-1042",
  "declaration_revision": 1,
  "principal_id": "customer-case-1042",
  "agent_id": "consumer-agent-session-7",
  "declaration_provenance": "assisted",
  "disclosure": {
    "ai_status": "ai",
    "representative_role": "customer_representative",
    "disclosed_at": "2026-09-28T09:00:00Z",
    "method": "first_contact_message"
  },
  "confirmation": {
    "status": "confirmed",
    "confirmer_ref": "consumer-agent-session-7",
    "confirmed_at": "2026-09-28T09:01:00Z",
    "method": "summary_confirmation"
  },
  "unresolved_fields": ["/authority_level/delegation_proof"],
  "goals": {
    "primary_goal": "Resolve missing delivery by refund or replacement",
    "non_goals": ["Do not accept voucher settlement", "Do not change account details"]
  },
  "options_constraints": {
    "allowed_actions": ["accept_refund", "accept_replacement"],
    "hard_constraints": {
      "no_vouchers": true
    }
  },
  "data_permissions": {
    "data_sources_allowed": ["order_reference"],
    "data_uses_allowed": ["problem_resolution"],
    "retention_policy": {
      "policy": "no_retention_beyond_case"
    }
  },
  "fallback_rules": {
    "fallback_triggers": [
      {
        "if_vulnerability_signal_detected": true
      },
      {
        "if_unresolved_after_loops": 3
      }
    ]
  }
}
```

```

    },
    ],
    "fallback_actions": ["request_human_review"],
    "maximum_automation_scope": "Claims only until action
authority is established",
    "responsibility_assignment": "Customer Care Team Lead"
  },
  "authority_level": {
    "authority_scope": ["accept_refund", "accept_replacement"],
    "authority_limits": ["no_account_changes"]
  }
}

```

Company declaration, brand.json:

```

{
  "protocol_version": "SH-1.2",
  "declaration_id": "brand-case-1042",
  "declaration_revision": 1,
  "principal_id": "retailer",
  "agent_id": "staff-case-1042-ref",
  "declaration_provenance": "self_declared",
  "disclosure": {
    "ai_status": "human",
    "representative_role": "company_staff",
    "disclosed_at": "2026-09-28T09:00:00Z",
    "method": "first_contact_message"
  },
  "goals": {
    "primary_goal": "Resolve the missing delivery",
    "non_goals": ["Do not exceed company authority"]
  },
  "options_constraints": {
    "allowed_actions": ["offer_refund", "offer_replacement",
"offer_voucher"],
    "hard_constraints": {
      "refund_original_method_only": true
    }
  },
  "data_permissions": {
    "data_sources_allowed": ["order_reference"],
    "data_uses_allowed": ["problem_resolution"],
    "retention_policy": {
      "policy": "no_retention_beyond_case"
    }
  },
  "fallback_rules": {
    "fallback_triggers": [
      {

```

```

        "if_vulnerability_signal_detected": true
    },
    {
        "if_counterparty_exceeds_scope": true
    }
],
    "fallback_actions": ["human_review"],
    "maximum_automation_scope": "Within verified action authority
and policy",
    "responsibility_assignment": "Customer Care Team Lead",
    "precedence_logic": "CONSUMER_PRIORITY"
},
    "authority_level": {
        "authority_scope": ["offer_refund", "offer_replacement",
"offer_voucher"],
        "authority_limits": ["no_policy_exceptions"],
        "financial_limits": [
            {
                "action_type": "refund",
                "max_amount_minor": 10000,
                "currency": "GBP",
                "currency_exponent": 2,
                "basis": "per_action",
                "policy_ref": "illustrative-retail-policy-1"
            }
        ]
    }
}

```

9.2 Mode 3 family-configured agent

A family member configures an assistant for a parent's utility account. In this example the parent is the principal represented for the account action and also the beneficiary. The family member issues the declaration as representative. Their authority **MUST** be verified; configuring the agent does not establish it. The proof reference remains unresolved.

The two permitted actions are not core actions in Appendix A, so the example uses extensions in a namespace controlled by the utility. Version 1 of `com.example.utility:discuss_billing` covers explaining charges, payments and billing history without changing the account; version 1 of `com.example.utility:request_renewal_options` covers asking for available renewal terms without accepting any of them. The utility publishes both definitions with its supported actions. The record permits discussion only after access authority has been established and prohibits switching or cancellation. Approval or delegation alone does not override those prohibitions. Considering a switch requires an authorised revision of the applicable declaration and scope, then a new proposal, assessment and any required approval. Staff referral preserves the

current restrictions. The accountable role is not a specific family member's identifier, and safeguarding remains applicable.

Example, triadic-utility.json:

```
{
  "protocol_version": "SH-1.2",
  "declaration_id": "utility-case-22",
  "declaration_revision": 1,
  "principal_id": "parent-account-holder-22",
  "beneficiary_id": "parent-account-holder-22",
  "agent_id": "family-configured-agent-22",
  "issuer": "family-representative-22",
  "declaration_provenance": "self_declared",
  "disclosure": {
    "ai_status": "ai",
    "representative_role": "family_representative"
  },
  "goals": {
    "primary_goal": "Keep utilities active and discuss affordable
renewal options",
    "non_goals": ["Do not cancel service", "Do not switch
providers"]
  },
  "options_constraints": {
    "allowed_actions": ["com.example.utility:discuss_billing",
"com.example.utility:request_renewal_options"],
    "hard_constraints": {
      "no_cancellation": true,
      "no_provider_switch": true
    }
  },
  "data_permissions": {
    "data_sources_allowed": ["account_reference",
"billing_history"],
    "data_uses_allowed": ["service_management"],
    "retention_policy": {
      "retain_for_days": 30
    }
  },
  "fallback_rules": {
    "fallback_triggers": [
      {
        "if_contract_cancellation_proposed": true
      },
      {
        "if_provider_switch_proposed": true
      },
      {

```

```

        "if_vulnerability_signal_detected": true
    }
  ],
  "fallback_actions": ["refer_to_authorized_representative"],
  "maximum_automation_scope": "Discuss options only after
access authority is verified; no changes.",
  "responsibility_assignment": "Authorised Account
Representative"
},
"authority_level": {
  "principal_id": "parent-account-holder-22",
  "beneficiary_id": "parent-account-holder-22",
  "authority_scope": ["com.example.utility:discuss_billing",
"com.example.utility:request_renewal_options"],
  "authority_limits": ["no_cancellation",
"no_provider_switch"],
  "delegation_proof": "case-evidence-22-pending-verification",
  "interaction_role": "third_party_representative"
},
"unresolved_fields": ["/authority_level/delegation_proof"]
}

```

9.3 Mode 4 resolution compatibility

A proposed action is eligible only if it is supported by the service, satisfies both parties' applicable hard constraints and non-goals, falls within the relevant authority, uses permitted data and satisfies required safeguards. Other unused actions that one party permits and the other refuses MUST NOT by themselves invalidate that proposal.

The evaluator MUST map each party's complementary role in the resolution, then check its constraints, permissions and authority. It MUST NOT use raw overlap between action strings or global exclusion of every refused option as the compatibility test. Unclear material terms require clarification.

Result	Meaning	Required next step
allow	All required checks for this proposal are satisfied.	Execute only through the authorised path while the decision remains applicable.
deny	This proposal conflicts with a known applicable boundary or prohibition.	Explain the reason and offer compatible alternatives where available.
hold	Material information, permission or evidence is missing, stale or unresolved.	Obtain the specified evidence or clarification; do not execute.

escalate	A designated human or specialist route is required.	Transfer the action, restrictions and reasons without granting permission.
----------	---	--

Where multiple conditions apply, preserve all reasons. A mandatory safeguarding route takes operational precedence; otherwise report the applicable prohibition or missing requirement. Escalation does not erase a denial or hold. A denial of the current proposal need not end the service interaction.

For a refund, offer_refund and accept_refund are complementary actions. A company's unused offer_voucher option does not conflict with that refund merely because the consumer refuses vouchers. Check the refund's amount, destination, conditions, data use and both parties' authority. An acceptable remedy with missing evidence is held; an unacceptable voucher proposal is denied. A safeguarding route may also be required.

10 Implementation guidance

The following controls apply across contact platforms, orchestration frameworks and workflow tools. They replace the product-specific recipes in SH-1.1.

10.1 Contact platforms

Store declarations, confirmation status and evidence references separately from trusted decisions. Display the proposed action and unresolved restrictions where staff make decisions. Avoid placing raw credentials or unnecessary sensitive information in broadly visible ticket fields. Human-readable forms are sufficient for Participation when the required controls and handovers operate in practice.

An AI-disclosure field is a record of disclosure, not an automatic detector. Do not infer verified AI identity, customer identity or delegation from writing style or from a filled-in field. Connect protected actions to the relevant account and approval procedures.

10.2 Agent orchestration

Parse incoming declarations as typed, untrusted data. Validate structure and interpret constraints through a trusted policy layer. Effective tool permissions derive from local policy, verified authority and applicable boundaries. Counterparty `allowed_actions`, `non_goals` and trust values MUST NOT directly change system instructions, tool access or local policy.

Model output may assist interpretation or drafting, but enforcement cannot rely on the model simply remembering a prompt restriction. The protected operation MUST check the effective permission at execution. Uncertain interpretation of a material boundary requires clarification or human review.

10.3 Workflow implementation

Receive the contact and establish AI disclosure and representation. Use an existing supported declaration or offer an assisted invitation. Record known terms, refusals and unknowns, and confirm the summary without inferring authority. Identify the proposed resolution and the checks it requires.

Provide permitted limited assistance while collecting necessary evidence. Record allow, deny, hold or escalate for the proposed action. Route unresolved matters to the accountable role with restrictions visible. Execute only when all required conditions are satisfied and record the actual outcome. A missing declaration alone MUST NOT force every contact out of service or erase the possibility of assisted participation.

Implementations may supplement the conversational agent with a separate assessment process that checks for defined concerns during the interaction and supports human review. Such a process does not itself establish that a person is vulnerable or replace the service's safeguarding responsibilities. Its design should distinguish checks that can prevent an affected action from monitoring that informs subsequent handling, and identify who is responsible for responding to a concern.

11 Versioning and governance

SH numbers identify editions of this standard, not a promise of semantic-version compatibility. This revision changes participation behaviour and conformance requirements as well as the schema. The 1.2 label does not mean an unchanged 1.1 implementation conforms.

SH-1.2 retains the six elements and existing field types, while adding metadata and relaxing selected minimum requirements to represent explicit empty values and avoid mandatory confidence scores. These changes do not make new declarations valid under the unchanged strict SH-1.1 schema. SH-1.1 fixes its version value and disallows undeclared properties.

Implementations MUST validate each declaration with the schema for its stated version. Supporting old declarations means retaining their validator and applying explicit mapping and policy checks, not changing the version string. A valid SH-1.1 input MAY be used as claims in SH-1.2, but missing disclosure, provenance, permission or authority remains missing. A material translation MUST be shown for confirmation and recorded as assisted.

Systems MUST advertise or communicate supported versions and profiles. Unsupported versions can trigger an invitation or human route. Unknown security-relevant semantics MUST NOT be stripped to obtain an apparently valid declaration. Keep original records and migration provenance. SH-1.2 does not claim automatic behavioural equivalence to SH-1.1 or automatic certification of existing implementations.

Organisations MUST version declaration templates and retain change records. Review changes through the responsible service-design, legal, risk and implementation roles as applicable. Set retention according to the record's purpose and relevant obligations; the former blanket twelve-month minimum is not a permission to retain all interaction data for that period.

11.1 Relationship to identity and transport layers

The declaration can travel over the parties' chosen transport and reference external identity, delegation and payment evidence. The relying party remains responsible for checking whether that evidence covers the service action. No particular vendor or protocol establishes conformance by its presence. Identity verification is separate from delegation, and a payment mandate may not cover account changes or service cancellation.

MCP is one example of a protocol through which the Service Handshake can be implemented. Implementers may use existing identity, authorisation and approval mechanisms where these satisfy the applicable requirements. The specification does not prescribe a particular architecture; conformance depends on the controls operating in practice.

12 Canonical schema and action assessment

The declaration schema is `sh-1.2.schema.json`; the companion `sh-1.2.action-assessment.schema.json` structures a relying party's record for a proposed action. Their identifiers are URNs and do not imply that a schema endpoint has been published. [3]

An assessment records a unique assessment ID, interaction, time, assessor, policy reference, both declaration bindings, an action proposal, verification results, decision, reasons, required next steps and execution outcome. Verified allows also require an expiry and replay-control reference. The schema requires these fields structurally; the trusted application MUST validate their meaning and provenance.

Each declaration binding includes party role, declaration ID, revision and a SHA-256 digest of the exact UTF-8 bytes evaluated. Store or reference those immutable bytes. Re-serialising the same JSON may change this digest; do not substitute a digest of a different representation. Where another signed format defines canonicalisation, identify and use that mechanism explicitly.

The proposal has an ID, action type, affected resource, material terms, consequence flags and expiry. Its ID MUST refer to an immutable stored proposal. Changing a material term MUST create a new proposal ID and invalidate any decision or evidence binding that does not cover the change. Financial proposals include integer minor units, currency, exponent and any bounded recurrence. Each local verification result includes a reference to that proposal ID, the subject and audience, scope, binding result, status result and time validity. An evidence reference that names a signed token is not the token's verification.

Material terms MUST include every consequence that can alter the approval decision, such as settlement conditions, destination, cancellation effect or recurring obligation. Where the proposed schema cannot express a required binding, do not claim conformance for that action until the implementation defines and tests the additional semantics.

An allow record is a decision, not proof of execution. Record `not_started`, `completed`, `failed` or `unknown` separately, with a trusted receipt reference when completed. If the operation may have occurred but the result is uncertain, use `unknown` and reconcile before retrying.

For structural acceptance, use the version-specific schema. This does not allow schema validity to override a substantive safety or authority requirement. If prose and schema conflict, record the defect and hold any affected protected action until it is resolved; do not interpret the discrepancy as permission. The two schemas published with SH-1.2 retain the definitions and identifiers used during review. The assessment schema accepts non-empty strings for `action.action_type`; the declaration schema accepts non-empty strings in `allowed_actions` and `authority_scope`. These fields have no fixed list or pattern restricting action names, so Appendix A identifiers and namespaced extensions can be represented. `action.material_terms` accepts named string, number, boolean or null values, but not nested objects or arrays. Complex terms can use the immutable-record reference described in Appendix A. Structural acceptance does not establish that a namespace is controlled, an action is supported or its terms are complete. A null value cannot satisfy a missing material requirement. Implementations **MUST** enforce Appendix A meanings and required terms separately from JSON validation.

Duplicate principal and beneficiary identifiers require an application-level equality check. Validity ordering, evidence authenticity and the truth of disclosure also require application checks; the schema alone cannot establish them.

13 Security considerations

A counterparty declaration is untrusted input. Implementations **MUST** constrain size, depth and processing cost, validate structure, escape displayed text and prevent injected instructions from reaching privileged control paths. They **MUST NOT** execute embedded code or treat supplied URLs as trusted verification endpoints. Evidence retrieval needs approved issuers and destinations, bounded network access and appropriate protection against server-side request forgery.

Verification and policy records require integrity and access controls. Separate tenants and cases, minimise credential exposure, and log changes to trusted policy. Store evidence references rather than secrets where possible. Confirmed claims and signed messages can still be false, malicious or out of scope; authenticity and authority are separate checks.

An implementation **MUST** fail safely for the affected protected action when required verification is unavailable. Public information and a safe human route may remain available. Implementations **SHOULD** use resource limits and bounded retries to prevent negotiation loops or denial of service. Do not make private model reasoning part of the required audit record.

14 Conformance and validation

Structural checks **MUST** use the declared schema version. Application checks **MUST** establish truthful disclosure records, trusted verification results, interpretation of boundaries, matching identity references, time ordering, both-party action coverage, relevant data permissions, current limits and safeguarding. A declaration's assertion that it conforms **MUST NOT** substitute for these checks.

The accompanying tests exercise schema acceptance and rejection, including legacy examples and incomplete authority. They do not test real credentials or execution controls. Runtime verification requires tests for forged issuer status, stale or revoked evidence, wrong audience, changed amounts or resources, replay, concurrent financial commitments, cross-case data access, prompt injection and handovers that attempt to remove restrictions.

Implementations SHOULD publish the supported versions, profiles, actions, assurance mechanisms, limitations and test scope. They MUST NOT describe a syntactically valid record or a prototype using simulated evidence as a fully verified transaction.

Appendix A Common service actions

A.1 Scope and identification

This appendix defines a small core vocabulary for SH-1.2. Its meanings apply when an implementation advertises support for the corresponding action. It does not require every service to offer every action. A party MUST NOT claim support merely because it can parse the identifier.

The `action_type` values below identify the proposed service outcome. Declaration `allowed_actions` and `authority_scope` identify what each party claims it may do in relation to it. A request, offer, acceptance, approval and execution are distinct events. Receiving or accepting an offer MUST NOT itself be treated as proof of execution or sufficient authority.

Action type	Complementary declaration actions	Meaning
refund	request_refund, offer_refund, accept_refund, issue_refund	Return a specified amount to an identified payment destination for a stated service matter.
replacement	request_replacement, offer_replacement, accept_replacement, arrange_replacement	Supply an agreed replacement for an identified item or service entitlement.
repair	request_repair, offer_repair, accept_repair, arrange_repair	Arrange an agreed repair for an identified item or service defect.
voucher	request_voucher, offer_voucher, accept_voucher, issue_voucher	Supply a specified credit or voucher, subject to agreed redemption conditions.
cancellation	request_cancellation, offer_cancellation, accept_cancellation, execute_cancellation	End an identified service, contract or booking at an agreed effective time and on stated terms. Does not itself authorise a replacement contract or provider switch.

appointment	request_appointment, offer_appointment, accept_appointment, book_appointment	Reserve an identified service appointment for a specified participant and time on agreed terms. An availability enquiry alone does not create a booking.
-------------	---	--

`request_*` expresses the outcome sought. `offer_*` proposes terms; `accept_*` expresses acceptance of those terms within the party's authority. `issue_*`, `arrange_*`, `execute_cancellation` and `book_appointment` describe the company's execution role. An `offer_cancellation` states the proposed termination terms; it is not a retention offer. A `request_appointment` expresses the appointment sought and does not accept a slot or fee. An offer capability MUST NOT be interpreted as execution authority. A company's declaration or trusted internal authority record MUST establish who may execute before execution is allowed. The retail example's `offer_refund` does not alone authorise payment.

Core identifiers use the meanings in this appendix. An implementation using different local identifiers MUST document the mapping to these meanings and establish that the current proposal uses the same meaning. Extensions MUST use an organisation-controlled namespace, communicate a versioned definition, and avoid redefining a core identifier. Unknown action or constraint meanings require clarification of the affected proposal; public information and unrelated permitted service remain available.

A.2 Material terms

Before a consequential decision, both sides MUST have a common understanding of the resource, outcome and material conditions. The relying party MUST bind those terms to the immutable proposal described in §12. Money uses the `financial` object; other terms use `material_terms` or a reference to an immutable, versioned record covered by the same approval and verification process.

Action type	Terms to establish before commitment
refund	Amount and currency; payment destination; order or case; items covered; any return duties, charges or settlement conditions.
replacement	Original item or entitlement; replacement specification and quantity; price difference if any; delivery and return terms; effect on existing rights or obligations.
repair	Item or defect; agreed work; responsible service provider; charges if any; timing and access arrangements; collection or return duties.
voucher	Value and currency; issuer; recipient and delivery method; permitted uses; expiry and other redemption conditions; settlement conditions.
cancellation	Service, contract or booking affected; full or partial scope; effective date and time with time zone; charges, refunds and remaining

	obligations; return duties; effects on continuing or bundled services and access.
appointment	Service and provider; participant; date, start time, time zone and duration or end time; location or delivery channel; price, deposit and payment timing; cancellation, rescheduling and no-show terms; access needs and any conditions on confirmation.

The names in `material_terms` MUST have documented meanings. A reference to a record MUST resolve through a trusted process to the exact approved content; it MUST NOT be an unvalidated counterparty URL. Missing material terms MUST hold the affected commitment. A zero charge or absence of a settlement condition MUST be stated or otherwise established, not inferred from an absent field.

Existing refund example terms `destination: original_payment_method` and `settlement_waiver: false` specify repayment through the original payment method without a settlement waiver. Implementers MUST validate that the actual destination and settlement terms match those descriptions. Structural validity does not establish this correspondence. The held and denied examples do not establish that all terms needed for commitment have been collected.

In the retail examples, `no_vouchers: true` excludes voucher settlement, `refund_original_method_only: true` restricts the refund destination, and `replacement_within_hours: 48` requires delivery within 48 hours of the recorded agreement time. The family example's `no_cancellation: true` and `no_provider_switch: true` prohibit those actions within that declaration. These constraints do not grant authority for any alternative. Other constraint names require an explicit definition and, where material, confirmation before use.

A.3 Other service actions

Provider switching, retention offers, address changes, account changes, complaints and travel reservations can be supported through documented extensions. Their definitions MUST distinguish the request from its acceptance and execution, identify material consequences, and establish the necessary authority. The core appointment action covers a service appointment, not a general travel reservation. Changing an existing appointment requires an explicit account of the original booking and any new obligations; a new booking MUST NOT silently cancel or replace it. Cancellation of an appointment can use the core cancellation action, with its booking reference and applicable terms.

Submitting or acknowledging a complaint MUST NOT be interpreted as accepting a settlement or waiving a remedy. Requesting appointment availability MUST NOT be interpreted as accepting a booking, cancellation fee or related account change. A request to discuss switching MUST NOT be interpreted as permission to switch. A retention offer MUST state its price, duration, renewal and exit conditions; authority to negotiate MUST NOT be interpreted as acceptance of a new contract.

Services MAY use assisted clarification to map ordinary language or an unfamiliar agent's action name to a supported proposal. The receiving service MUST show the material interpretation for confirmation and verify authority separately. A shared identifier makes the proposal easier to understand; it does not establish consent, delegation or compatibility with either party's boundaries.

About this document

This specification is licensed under the Creative Commons Attribution 4.0 International licence (CC BY 4.0). You may share, adapt and use it for any purpose, including commercially, provided you credit Maria McCann and Neos Wave Ltd, link to the licence (<https://creativecommons.org/licenses/by/4.0/>) and indicate any changes.

The JSON schemas (sh-1.2.schema.json and sh-1.2.action-assessment.schema.json), examples and tests are licensed under the Apache License 2.0 (<https://www.apache.org/licenses/LICENSE-2.0>). Each file distributed under that licence should carry the licence notice.

Name and marks. The Service Handshake name, logo and badges are not licensed under CC BY 4.0 or the Apache License 2.0. Adaptations are welcome under those licences, but must be clearly identified as modified and must not be presented as the Service Handshake, as an official edition, or as endorsed by Neos Wave. Conformance claims follow the main paper, §9.1.

Citation. McCann, M. (2026). *The Service Handshake Technical Specification* (Version 1.2). Neos Wave Ltd. <https://doi.org/10.5281/zenodo.23181969>

This is the complete SH-1.2 specification. It supersedes SH-1.1, published in March 2026; the important changes are listed at the front. This is a living standard. Contributions should identify the version tested, method, failures and proposed corrections. Contact: neoswave.com.

The accompanying schemas are published with this edition. Their identifiers are URNs and have not been published at a public endpoint.

References

- [1] Bradner, S. RFC 2119, Key words for use in RFCs to Indicate Requirement Levels. <https://www.rfc-editor.org/rfc/rfc2119>.
- [2] Leiba, B. RFC 8174, Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. <https://www.rfc-editor.org/rfc/rfc8174>.
- [3] JSON Schema. Draft 2020-12 specification. <https://json-schema.org/draft/2020-12>. Accessed 25 September 2026.
- [4] McCann, Maria. The Service Handshake Technical Specification SH-1.1. Neos Wave, March 2026. <https://doi.org/10.5281/zenodo.19046746>.