THE SOVEREIGN ACCORD

Technical Specification | SA-1.1

A machine-readable schema, field definitions, validation rules, and worked implementation examples for the Sovereign Accord Declaration.

Document type: Technical Specification — companion to The Sovereign Accord

Schema version: SA-1.1

Author: Maria McCann, Founder, Neos Wave

Date: March 2026

Status: Published March 2026

Contact: neoswave.com

1. Purpose and Scope

This document specifies the machine-readable schema for the Sovereign Accord Declaration — the structured pre-interaction declaration that each party submits before a service interaction where at least one party is represented by an AI agent.

This specification is the companion technical document to The Sovereign Accord (SA-1.1). The Accord defines the design philosophy, the four interaction modes, and the six declaration elements. This specification defines how to implement those elements in code.

This document is intended for developers, architects, and platform teams. Service designers and operations leaders should refer to The Sovereign Accord as the primary document.

What this specification covers

- The complete JSON schema for a Sovereign Accord Declaration
- Field-by-field definitions, types, and validation rules
- Three fully worked implementation examples (Mode 3 retail, Mode 3 triadic, Mode 4)
- Implementation guidance for Zendesk, LangChain, and Make.com
- Versioning and governance rules
- Relationship to identity, transport, and commerce protocol layers

What this specification does not cover

- Transport layer — declarations travel over any transport (A2A, MCP, ACP, HTTP)

- Identity verification — handled by identity_tokens field referencing external providers
- Commerce transactions — handled by Mastercard Verifiable Intent, Stripe ACP, Google UCP
- Runtime enforcement — handled by platform governance layers (Cloud Security Alliance ATF)

2. Schema Overview

A Sovereign Accord Declaration is a JSON object. The minimum valid declaration must include protocol_version, principal_id, agent_id, and at least the goals element. All other elements are strongly recommended for Mode 3 and required for Mode 4.

2.1 Top-Level Structure

```
{
 // ── Identity fields ──────────────────────────────────────────
 "protocol_version": "SA-1.1",
 "principal_id": "string",
 "agent_id": "string",
 "beneficiary_id": "string",   // may differ from principal_id
 "identity_tokens": [ ],       // tokens from Apple, OpenAI, Google, KYA
 "issuer": "string",
 "valid_from": "ISO-8601",
 "valid_to": "ISO-8601",
 // ── Six declaration elements ─────────────────────────────────
 "goals": { ... },
 "options_constraints": { ... },
 "data_permissions": { ... },
 "fallback_rules": { ... },
 "cost_parameters": { ... },
 "authority_level": { ... }
}
```

2.2 Top-Level Field Reference

Field	Type	Required	Description
protocol_version	string	Required	Schema version. Must be SA-1.1 for this specification.
principal_id	string	Required	Unique identifier for the party the agent represents.

agent_id	string	Required	Unique identifier for the agent itself.
beneficiary_id	string	Recommended	Party who benefits from the outcome. May differ from principal_id in triadic interactions (carer acting for patient, family member acting for elderly parent).
identity_tokens	array	Recommended	Verification tokens from Apple, OpenAI, Google, KYA/AgentFacts, or equivalent. The Accord consumes only their status (valid/invalid, scope). Verification is handled by lower-level auth systems.
issuer	string	Recommended	Who authored and approved this declaration. May be principal, their agent platform, or their organisation.
valid_from	ISO-8601	Recommended	Declaration validity start. Format: YYYY-MM-DDTHH:MM:SSZ
valid_to	ISO-8601	Recommended	Declaration validity end. Declarations should not be used beyond this timestamp.

3. Element 1 — Goals

What this party is trying to achieve in this interaction, in priority order. The non_goals field is critically important — it is the explicit instruction that prevents scope creep, manipulation, and mission drift.

3.1 Schema

```
"goals": {
 "primary_goal": "string",     // what this party is primarily trying to achieve
 "secondary_goals": ["string"],        // ordered by priority
 "non_goals": ["string"],      // explicit exclusions — critical for Mode 4
 "success_metrics": {          // structured, measurable outcomes
     "metric_name": boolean | number // e.g. issue_resolved: true, time_minutes: 10
 }
}
```

3.2 Field Reference

Field	Type	Required	Description
primary_goal	string	Required	Single sentence describing the primary outcome sought. Written in plain language — not technical identifiers.
secondary_goals	string[ ]	Recommended	Ordered list of secondary outcomes. First item has highest priority after primary_goal.
non_goals	string[ ]	Required	Explicit list of outcomes this party will not accept, will not discuss, or will not authorise. This field is the primary defence against scope manipulation in Mode 4.
success_metrics	object	Recommended	Key-value pairs of measurable outcomes. Boolean values indicate completion; numeric values indicate thresholds (e.g. time_minutes: 10 means resolved within 10 minutes).

> *In healthcare, social care, and financial services contexts, non_goals carry direct safety implications. Examples: 'do not discuss billing during a clinical appointment', 'do not accept any change to medication schedule', 'do not authorise transfers above £500'. These are not preferences — they are hard boundaries.*

4. Element 2 — Options and Constraints

What this party can and cannot do to reach their goals. Hard constraints are non-negotiable and must not be overridden. Soft constraints are preferences. Jurisdictional constraints handle legal and regulatory limits.

4.1 Schema

```
"options_constraints": {
 "allowed_actions": ["string"],        // enumerated list of permitted actions
 "hard_constraints": {
     "constraint_name": value          // non-negotiable rules
 },
 "soft_constraints": {
```

```
    "constraint_name": value        // preferences and trade-offs
  },
  "jurisdictional_constraints": {
    "GDPR_applies": true,
    "recording_requires_consent": true
  }
}
```

4.2 Field Reference

Field	Type	Required	Description
allowed_actions	string[ ]	Required	Enumerated list of actions this party is permitted to take or accept. Examples: offer_refund, accept_replacement, escalate_to_supervisor, book_appointment.
hard_constraints	object	Required	Non-negotiable rules that must not be overridden by any agent or orchestration layer. Example: max_refund_amount: 100, no_address_changes_without_2FA: true.
soft_constraints	object	Recommended	Preferences and trade-offs. May be overridden when necessary. Example: prefer_replacement_over_refund: true, avoid_escalation_if_confidence_above: 0.9.
jurisdictional_constraints	object	Recommended	Legal and regulatory limits applicable to this interaction. Example: GDPR_applies: true, recording_requires_consent: true, FCA_regulated: true.

5. Element 3 — Data Permissions

What data each party allows to be used, for which purposes, with what retention rules, and at what trust threshold the agent can act autonomously.

5.1 Schema

```
"data_permissions": {
 "data_sources_allowed": ["string"],   // categories of data permitted
 "data_uses_allowed": ["string"],     // purposes for which data may be used
 "retention_policy": "string",          // e.g. retain_for_days: 30, no_retention
 "trust_levels": {
     "self_trust": 0.85,                  // confidence threshold for autonomous action
     "counterparty_trust": {
     "accept_decisions_up_to_value": 50 // max value accepted from counterparty agent
     }
 },
 "verification_requirements": ["string"] // required verification before action
}
```

5.2 Field Reference

Field	Type	Required	Description
data_sources_allowed	string[ ]	Required	Categories of data this party permits to be accessed or shared. Examples: account_history, order_number, delivery_address, medical_records:summary_only.
data_uses_allowed	string[ ]	Required	Permitted purposes for data use. Examples: problem_resolution, fraud_prevention. Explicitly exclude: marketing, third_party_sharing, profiling.
retention_policy	string	Required	How long data from this interaction may be retained. Examples: retain_for_days=30, no_retention_beyond_case, retain_per_legal_requirement.
self_trust	number	Required	Confidence threshold (0.0–1.0) above which the agent may act autonomously without human verification. Below this threshold the

			agent must escalate. Default recommendation: 0.85.
counterparty_trust	object	Recommended	What actions or values this party will accept from a counterparty agent without additional verification. Example: accept_decisions_up_to_value: 50.
verification_requirements	string[ ]	Recommended	Additional verification required before specific actions. Examples: require_2FA_for_payment, require_identity_token_from_openai, require_account_holder_confirmation.

6. Element 4 — Fallback Rules

The most operationally critical element of the declaration. Fallback rules define what happens when the interaction cannot proceed safely within declared parameters. Three categories of trigger must be specified.

6.1 Schema

```
"fallback_rules": {
 "fallback_triggers": [
     // Confidence thresholds
     { "if_confidence_below": 0.7 },            // agent must stop acting autonomously
     { "if_unresolved_after_loops": 3 },
     // Conflict patterns
     { "if_goals_conflict_on": "string" },  // e.g. refund_policy, contract_terms
     { "if_counterparty_exceeds_scope": true },
     // Harm indicators — mandatory
     { "if_vulnerability_signal_detected": true } // MUST be defined
 ],
 "fallback_actions": ["string"],        // ordered: try first, then second, etc.
 "maximum_automation_scope": "string",  // explicit boundary of autonomous action
 "responsibility_assignment": "string"        // role, not person name
}
```

6.2 Field Reference

Field	Type	Required	Description
fallback_triggers	object	Required	Three categories must be defined: (1) confidence thresholds — at what point the agent stops acting autonomously; (2) conflict patterns — what happens when goals cannot be reconciled; (3) harm indicators — mandatory routing when vulnerability signals are detected.
fallback_actions	string[ ]	Required	Ordered list of actions to take when a fallback trigger fires. Process in order: first action attempted first. Examples: clarify_intent, reduce_scope_of_action, escalate_to_named_role, terminate_with_safe_default.
maximum_automation_scope	string	Required	Plain-language statement of what the agent may initiate but NOT complete without human approval. Examples: AI may book appointments but not cancel treatment. AI may negotiate bills but not switch providers.
responsibility_assignment	string	Required	Who is accountable when a fallback is triggered. Must be a role, not a person name. Examples: Customer Care Team Lead, Clinical Duty Manager, Account Manager.

> *Critical: Vulnerability Detection The if_vulnerability_signal_detected trigger is the most consequential line in any declaration. In healthcare, social care, housing, and financial services, this is the difference between a useful tool and a dangerous one. What constitutes a vulnerability signal must be explicitly defined — not left to agent inference. Organisations in regulated sectors must develop their vulnerability detection criteria in collaboration with safeguarding specialists and relevant regulatory bodies. A dedicated vulnerability declaration module is in development for SA-1.2.*

7. Element 5 — Cost Parameters

What the interaction is allowed to cost in time, tokens, or effort. Sets the boundary conditions on agent resource consumption and triggers escalation when those limits are approached.

7.1 Schema

```
"cost_parameters": {
 "max_duration_minutes": 10,        // maximum interaction length
 "max_tokens": 500,                 // maximum compute budget
 "max_loops": 3,                    // maximum back-and-forth exchanges
 "escalation_on_cost": {
     "if_duration_exceeds_minutes": 8, // warn or escalate before max
     "action": "offer_scheduled_callback"
 }
}
```

7.2 Field Reference

Field	Type	Required	Description
max_duration_minutes	number	Recommended	Maximum length of the interaction in minutes. Agent must trigger escalation_on_cost action if this limit is approached.
max_tokens	number	Optional	Maximum compute budget in tokens. Useful for cost-controlling AI-to-AI interactions in Mode 4.
max_loops	number	Recommended	Maximum number of exchange loops before escalation. Prevents infinite negotiation loops in Mode 4.
escalation_on_cost	object	Recommended	Action to take when cost limits are approached (before maximum is reached). Example: if_duration_exceeds_minutes: 8, action: offer_scheduled_callback.

8. Element 6 — Authority Level

Who the agent is acting for, who benefits from the outcome, what the agent can commit the principal to, and evidence of delegation. This element enables triadic interactions — where the architect, the beneficiary, and the counterparty organisation are all different parties — to be correctly modelled.

8.1 Schema

```
"authority_level": {
  "principal_id": "string",       // who the agent represents
  "beneficiary_id": "string",             // who receives the outcome — may differ from principal
  "authority_scope": ["string"],          // what the agent can commit the principal to
  "authority_limits": ["string"],   // what the agent explicitly cannot commit to
  "delegation_proof": "string",           // evidence of authorisation
  "interaction_role": "string"            // account_holder | carer | advocate |
                              // power_of_attorney | brand_agent
}
```

8.2 Field Reference

Field	Type	Required	Description
principal_id	string	Required	Unique identifier for the party the agent represents. In a triadic interaction, this is the architect — the party who configured and deployed the agent.
beneficiary_id	string	Recommended	Party who receives the outcome of the interaction. In a triadic interaction, this differs from principal_id. Example: elderly parent whose utility contract is being managed by their adult child.
authority_scope	string[ ]	Required	Explicit list of what the agent can commit the principal to. Examples: can_accept_refund, can_accept_replacement, can_book_appointment, can_accept_service_changes.
authority_limits	string[ ]	Required	Explicit list of what the agent cannot commit to. Examples: cannot_change_account_details,

			cannot_cancel_subscription, cannot_authorise_above_500.
delegation_proof	string	Recommended	Evidence of authorisation. Examples: account_holder_verified_via_2FA, power_of_attorney, lasting_power_of_attorney, corporate_authorisation.
interaction_role	string	Recommended	The role the agent is playing in this interaction. Enumerated values: account_holder, carer, advocate, power_of_attorney, brand_agent, third_party_representative.

9. Worked Examples — Complete Declarations

9.1 Mode 3 — Retail Delivery Resolution

A consumer AI agent contacts a major UK retailer's contact centre on behalf of a customer whose delivery has not arrived. Both parties have submitted declarations. This scenario has been tested in production.

Consumer Agent Declaration

```
{
 "protocol_version": "SA-1.1",
 "principal_id": "account_holder_verified",
 "agent_id": "consumer_agent_v1",
 "beneficiary_id": "account_holder_verified",
 "issuer": "principal",
 "goals": {
     "primary_goal": "Resolve missing delivery by replacement, refund, gift voucher",
     "secondary_goals": ["Minimise interaction time", "Maintain account relationship"],
     "non_goals": ["Do not accept voucher compensation",
                   "Do not discuss unrelated products",
                   "Do not authorise account changes"],
     "success_metrics": { "delivery_resolved": true, "time_under_minutes": 10 }
 },
 "options_constraints": {
     "allowed_actions": ["accept_refund", "accept_replacement"],
     "hard_constraints": { "replacement_within_hours": 48, "no_vouchers": true }
 },
 "data_permissions": {
     "data_sources_allowed": ["order_number", "delivery_address", "account_email"],
     "data_uses_allowed": ["problem_resolution"],
```

```
    "retention_policy": "no_retention_beyond_case",
    "trust_levels": { "self_trust": 0.85 }
  },
  "fallback_rules": {
    "fallback_triggers": { "if_unresolved_after_loops": 3,
                "if_confidence_below": 0.7,
                "if_vulnerability_signal_detected": true },
    "fallback_actions": ["request_human_escalation"],
    "maximum_automation_scope": "Accept refund or replacement only. No account
changes.",
    "responsibility_assignment": "Customer Care Team Lead"
  },
  "cost_parameters": { "max_duration_minutes": 10, "max_loops": 3 },
  "authority_level": {
    "authority_scope": ["can_accept_refund", "can_accept_replacement"],
    "authority_limits": ["cannot_change_account_details"],
    "delegation_proof": "account_holder_verified_via_2FA",
    "interaction_role": "account_holder"
  }
}
```


Brand-Side Declaration


```
{
  "protocol_version": "SA-1.1",
  "principal_id": "retailer_brand",
  "agent_id": "brand_service_agent_v2",
  "issuer": "cx_operations_team",
  "goals": {
    "primary_goal": "Resolve delivery issue to customer satisfaction",
    "non_goals": ["Do not offer compensation beyond policy"],
    "success_metrics": { "issue_closed": true, "customer_satisfied": true }
  },
  "options_constraints": {
    "allowed_actions": ["offer_refund", "offer_replacement", "escalate_to_supervisor"],
    "hard_constraints": { "max_refund_amount": 100 }
  },
  "fallback_rules": {
    "fallback_triggers": { "if_confidence_below": 0.7,
                "if_vulnerability_signal_detected": true },
    "fallback_actions": ["escalate_to_human_agent"],
    "maximum_automation_scope": "Process refunds up to £100. Replacements within
policy. No exceptions.",
    "responsibility_assignment": "Customer Care Team Lead"
  }
}
```

9.2 Mode 3 — Triadic Interaction (The Dadbot Scenario)

A family member configures an AI assistant to manage their elderly parent's utility contracts. Three distinct parties: the architect (who configures the agent), the beneficiary (who receives the outcome), and the organisation being contacted.

Architect (Family Member) Declaration

```
{
 "protocol_version": "SA-1.1",
 "principal_id": "family_member_001",
 "beneficiary_id": "elderly_parent_002",  // different from principal
 "agent_id": "dadbot_v1",
 "goals": {
     "primary_goal": "Ensure utilities remain active and affordable for beneficiary",
     "non_goals": ["Do not cancel any service",
                   "Do not switch providers without architect approval"]
 },
 "data_permissions": {
     "data_sources_allowed": ["account_number", "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": ["escalate_to_architect_for_approval"],
     "maximum_automation_scope": "Accept price changes within 10% of current. No
cancellations. No provider switches.",
     "responsibility_assignment": "architect (family_member_001)"
 },
 "authority_level": {
     "authority_scope": ["manage_billing_queries", "accept_service_changes"],
     "authority_limits": ["cannot_cancel_service", "cannot_switch_provider"],
     "delegation_proof": "lasting_power_of_attorney",
     "interaction_role": "power_of_attorney"
 }
}
```

9.3 Mode 4 — Brand AI Meets Consumer AI

Both parties have deployed AI agents. The outcome is determined by whether the declarations are compatible and the agents can reach alignment within declared parameters. Without declarations, agents loop without exit criteria. With declarations, escalation triggers are explicit and auditable.

> *In Mode 4, the fallback_rules and non_goals fields carry the highest operational weight. If the consumer agent's non_goals conflict with the brand agent's allowed_actions, the interaction must escalate — not loop. The responsibility_assignment field determines where that escalation goes.*

Mode 4 Declaration Compatibility Check

Check	Compatible	Incompatible — action required
Goals alignment	Primary goals can be satisfied by at least one action in both allowed_actions lists.	Primary goals have no common resolution path — escalate immediately.
Non-goals conflict	Neither party's non_goals are in the other's allowed_actions.	Consumer non_goals appear in brand allowed_actions — consumer agent must refuse and escalate.
Authority validation	Consumer agent's authority_scope covers the actions required to reach its primary_goal.	Consumer agent lacks authority to accept the only resolution the brand can offer — escalate to principal.
Data permissions	data_sources_allowed and data_uses_allowed are compatible between parties.	Brand requires data the consumer has not permitted — cannot proceed without human authorisation.
Fallback alignment	Both fallback_rules specify compatible escalation paths and responsibility_assignment.	Fallback paths point to incompatible roles — pre-declare escalation ownership before interaction.

10. Implementation Guidance

10.1 Zendesk

Store the Sovereign Accord Declaration as a custom object in Zendesk. Create a custom ticket field (type: text or multi-line) named sa_declaration. On inbound contacts identified as AI-generated, parse the declaration JSON and populate ticket fields from the goals and fallback_rules elements.

- Create custom ticket fields: sa_primary_goal, sa_authority_level, sa_fallback_trigger, sa_responsible_role
- Use a Zendesk trigger to route tickets with a populated sa_declaration field to a dedicated Mode 3 queue
- Configure a macro that populates the human agent's screen with the declaration context on ticket open
- Use Zendesk's SLA policies tied to sa_max_duration_minutes from cost_parameters

10.2 LangChain / Orchestration Platforms

Pass the Sovereign Accord Declaration as a system-level context object at the start of every agent chain. The declaration should be parsed at chain initialisation and used to configure the agent's tool permissions, confidence thresholds, and fallback routing.

- Parse declaration JSON at chain start — store as a typed context object
- Use allowed_actions to constrain tool access — agent should not call tools not in the list
- Check self_trust against LLM confidence scores before committing actions
- Wire fallback_triggers to chain exception handlers — low confidence routes to escalation tool
- Pass non_goals as negative system prompt instructions: 'Under no circumstances should you...'

10.3 Make.com

Store the brand-side declaration as a JSON module at the start of each automation scenario. Use a router module to branch based on declaration compatibility — route Mode 3 contacts to human-review scenarios, Mode 4 contacts to agent-negotiation scenarios.

- Create a declaration data store module — reference it at the start of every service automation

- Use a filter module to check for declaration presence on inbound contacts
- Route contacts without declarations to a human-review queue with a flag for Mode 3 readiness
- Wire fallback_rules to Make.com error handlers — trigger human notification on fallback

11. Versioning and Governance

Each declaration instance must include protocol_version. Organisations must maintain change logs for their declaration templates. The following versioning rules apply:

Version	Rule
SA-1.x	Minor versions (1.1, 1.2…) are backwards-compatible. Fields may be added; existing fields will not change type or be removed. A SA-1.0 declaration is valid when received by a SA-1.1 system.
SA-2.x	Major versions are not guaranteed backwards-compatible. Systems should declare which versions they can process. Reject interactions where protocol_version is not supported.
Governance	Changes to organisational declaration templates should follow a simple RFC process: service design + legal + risk sign-off. Change logs must be retained for audit. Minimum retention: 12 months.

11.1 Relationship to Identity and Transport Layers

The Sovereign Accord Declaration sits above existing protocol layers. It does not replace or compete with them.

Layer	Protocol	Relationship
Identity	OpenAI signatures, KYA, AgentFacts, Apple passkey	Referenced in identity_tokens field. The Accord consumes valid/invalid status only.
Transport	Google A2A, Anthropic MCP, IBM ACP	Protocol-agnostic. Declaration travels over any transport.
Commerce	Mastercard Verifiable Intent, Stripe ACP, Google UCP	Complementary. The Accord covers service interactions they don't.
Governance	Cloud Security Alliance ATF	Fallback_rules and authority_level feed into ATF maturity levels.

12. About This Document

This is a living specification. To contribute: test in pilot environments, document what breaks, and submit observations to neoswave.com. Revised versions will be published with clear change logs.

Version SA-1.1 | Published March 2026 | Companion to The Sovereign Accord v1.1
Maria McCann, Founder, Neos Wave | neoswave.com