Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 1 June 2026

M. Gaikwad
Independent Researcher

28 November 2025

SOUTH: Stochastic Authorization for Agent and Service Requests
draft-gaikwad-south-authorization-00

Abstract

SOUTH defines an authorization protocol for evaluating requests issued by users, services, or autonomous agents. SOUTH allows a server to return a deterministic decision or an allow decision that is issued with a probability determined by local policy. This enables servers to incorporate uncertainty, contextual information, and load conditions into authorization decisions.

SOUTH is transport independent and can be composed with existing authentication mechanisms such as OAuth, OpenID Connect, mutual TLS, or DPoP. This document describes the request and response objects, decision semantics, and an HTTP binding for interoperable deployment.

Status of This Memo

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

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

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

This Internet-Draft will expire on 1 June 2026.

Copyright Notice

Table of Contents

1. Introduction

Systems that act on behalf of users or other services must decide whether a requested operation is permitted. Traditional authorization mechanisms rely on deterministic evaluation of attributes, roles, or policy rules. These mechanisms are effective in constrained environments, but they do not represent uncertainty, context variation, or adaptive behavior that arise in modern agent driven systems.

SOUTH defines a small authorization protocol that allows a server to return either a deterministic decision or a probabilistic allow decision. A server evaluates a structured request, computes a preference score, and maps this score to a decision. SOUTH also supports retry and defer outcomes that allow a server to signal transient conditions, such as load or missing context. SOUTH does not replace authentication or identity. Instead, it complements existing identity systems by defining a consistent authorization interface.

The name SOUTH is derived from "Stochastic Auth." It reflects the protocol’s ability to express authorization decisions that may include probabilistic behavior, allowing servers to map preference scores or contextual signals to non-deterministic outcomes when required by policy.

The protocol is transport independent. It can be used with HTTP, message queues, or local control interfaces. It can also be paired with authentication and integrity mechanisms such as OAuth based flows, OpenID Connect, mutual TLS, or DPoP proof of possession. SOUTH does not define how preference scores are computed. It defines the request and response structures and the semantics of decisions returned by a server.

SOUTH can be used by user agents, machine learning agents, service meshes, schedulers, MTAs, SDN controllers, and similar components that require an explicit authorization decision for a proposed action. This document describes terminology, message formats, decision semantics, transport bindings, and operational considerations for interoperable use.

2. Terminology

The terms defined in this section are used throughout this document. These definitions are specific to the SOUTH protocol.

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

Client The entity that sends a SOUTH request. A client may be a user agent, an automated agent, a service, a scheduler, or any component that requires an authorization decision for a proposed operation.

Server The entity that evaluates a SOUTH request and returns a decision. A server applies local policy to determine whether a request should be allowed, denied, retried, or deferred.

Request A structured object that describes the operation the client wishes to perform. A request includes an action, a target, and optional context that assists the server in making a decision.

Decision The outcome returned by a SOUTH server. Decisions include allow, deny, retry, defer, and error. SOUTH does not assign meaning to application level outcomes beyond these states.

Preference Score A numerical value in the interval [0, 1] that reflects how compatible a request is with server policy. SOUTH does not specify how the score is computed. The server maps the score to a decision.

Stochastic Allow A mode where the server returns an allow decision with a probability derived from the preference score. The exact mapping is defined by server policy.

Context Attachment Additional information supplied by the client or collected by the server that provides signals relevant to the decision. Examples include rate information, environmental data, recent behavior, or device posture. Context attachment formats are implementation dependent.

Integrity Binding A method used to ensure that the request has not been modified and that the sender can prove possession of a key. SOUTH is transport independent and can be paired with mechanisms such as mutual TLS or DPoP.

3. Motivation

Authorization mechanisms used in current systems are largely deterministic. A request is evaluated against a set of rules or attributes, and the server returns an allow or deny outcome. These mechanisms work well when policies are simple and when the environment is stable. Many modern deployments no longer meet these conditions.

Agent driven systems, service meshes, serverless frameworks, and automated workflows issue requests whose safety depends on transient information such as current load, recent behavior, signal quality, or incomplete state. A deterministic policy cannot represent these conditions without either becoming overly restrictive or overly permissive. Systems often fall back on large sets of static rules that do not scale and are difficult to maintain.

SOUTH introduces a small protocol that allows a server to encode preference and uncertainty directly. A server evaluates a structured request, computes a score, and maps this score to a decision. The server may return a deterministic allow or deny decision or may choose a probabilistic allow when the score falls in an uncertain range. SOUTH also supports explicit retry and defer outcomes so that clients can distinguish between a stable deny and a temporary condition.

This approach provides two benefits. First, it allows a server to represent policies that depend on contextual or uncertain information without expanding the policy surface. Second, it gives clients clear signals about whether a denial is final or whether a request may succeed after obtaining more context or waiting for transient conditions to change.

SOUTH is intended for systems that require lightweight authorization decisions between components that operate at high frequency or under shifting operational context. This aligns with zero trust principles [NIST-ZT]. It can be used by user agents, machine learning agents, schedulers, MTAs, SDN controllers, and other components that need explicit authorization semantics for proposed operations.

4. Design Goals and Non-Goals

This section states the main design goals for SOUTH and clarifies which problems are outside the scope of this document.

4.1. Design Goals

SOUTH is designed with the following goals.

* *Simple decision interface.* SOUTH provides a small set of decision outcomes such as allow, deny, retry, and defer. The protocol does not try to encode full policy logic on the wire. Policy remains on the server.

* *Stochastic authorization.* SOUTH allows a server to map internal preference scores to probabilistic allow decisions. This supports policies that depend on uncertain signals or that aim to reduce predictability for probing clients.

* *Transport independence.* SOUTH defines request and response objects that are not tied to one specific transport. An HTTP binding is described in this document, but other bindings can be defined by deployment specific profiles.

* *Compatibility with existing identity.* SOUTH is intended to work alongside existing authentication and identity systems, such as OAuth based flows, OpenID Connect, mutual TLS, and DPoP. It does not define its own identity model.

* *Low integration cost.* SOUTH can be called by existing components as an external decision service. A client can construct a request object, send it to a SOUTH server, and apply the returned decision without changes to its internal policy representation.

* *Clear denial semantics.* SOUTH distinguishes between a stable deny, a temporary condition, and a request that should be retried or stepped up. This allows clients to react appropriately to authorization outcomes.

4.2. Non-Goals

SOUTH does not attempt to solve the following problems.

* *Authentication and identity.* SOUTH does not define how clients or servers authenticate. It assumes that deployments use existing mechanisms such as mutual TLS, OAuth, OpenID Connect, or similar protocols.

* *Policy language definition.* SOUTH does not define a policy language or a rule syntax. It defines only the interface for exchanging decisions. Policy representation and evaluation are local choices.

* *Data plane enforcement.* SOUTH does not specify how decisions are enforced in data planes, such as packet forwarding hardware or message brokers. Enforcement mechanisms are outside the scope of this document.

* *Global consistency.* SOUTH does not attempt to provide globally consistent authorization views across multiple servers. Any replication or coordination of policy state is the responsibility of the deployment.

* *End to end security proofs.* SOUTH is a building block. It does not provide a complete security framework for all authorization use cases and does not claim end to end security guarantees by itself.

5. Architecture

SOUTH is designed as a decision service that can be called by different types of clients. The protocol defines a logical client role, a server role, and supporting components such as policy stores and identity providers. SOUTH does not constrain how these roles are deployed or how they are mapped to physical systems.

5.1. Roles

The SOUTH architecture includes the following logical roles.

Client The client constructs a SOUTH request, sends it to a server, and applies the returned decision. Clients may be user agents, automated agents, application services, schedulers, mail transfer agents, or network controllers.

Server The server receives SOUTH requests, evaluates them against local policy and context, and returns decisions. A server may be implemented as a standalone service, as a component within a larger system, or as a library linked into an application.

Policy Store A policy store holds configuration and rules that the server uses to evaluate requests. The format and location of the policy store are not specified by SOUTH. A policy store may be a configuration file, a database, or an external policy engine.

Identity Provider An identity provider issues credentials or tokens that clients use to authenticate to the server. SOUTH does not define identity provider behavior but assumes that servers can validate client identities using existing mechanisms.

5.2. Interaction Model

The basic interaction pattern in SOUTH consists of a single request followed by a single response.

1. The client constructs a request object that describes the proposed operation. The request includes information about the acting subject, the action to be performed, the target object, and optional context.

2. The client sends the request to the server using a chosen transport. An HTTP binding using JSON is described in this document. Other bindings may be defined in separate documents.

3. The server evaluates the request according to local policy. The server may consult its policy store, fetch additional context, or apply internal scoring models.

4. The server returns a response that contains a decision and optional metadata. The decision indicates whether the operation is allowed, denied, should be retried, or should be deferred. The response may include a probability or score for diagnostic or logging purposes.

5. The client applies the decision. For example, a client may proceed with the operation, block it, queue it for later retry, or prompt for additional verification, depending on local behavior.

SOUTH does not specify how many requests a client may send or how servers handle concurrent requests. These aspects are deployment choices. Implementations are expected to apply standard practices for rate limiting, timeouts, and failure handling.

5.3. Deployment Patterns

SOUTH can be deployed in several patterns. This document does not mandate any specific topology, but the following patterns are expected to be common.

* *Central decision service.* A dedicated SOUTH server receives requests from many clients and applies shared policy. This is suitable for environments that want a single authorization view across multiple systems.

* *Sidecar or local proxy.* A SOUTH server runs next to an application component and serves only that component. This avoids network hops but keeps policy evaluation separate from application logic.

* *Library integration.* SOUTH evaluation is implemented as a library linked into a client or gateway. The wire protocol may still be used for audit, testing, or cross language integration, but is not required for every decision.

* *Controller integration.* A control plane component, such as a service mesh controller, scheduler, or SDN controller, calls SOUTH during its own decision flow. The controller then programs local enforcement points based on the SOUTH decision.

In all patterns, SOUTH is intended to run in the control plane or management plane. Enforcement in data planes, such as packet forwarding paths or storage engines, is outside the scope of the protocol. Implementations are expected to map SOUTH decisions to local enforcement mechanisms.

6. Decision Semantics

SOUTH defines a small set of decision outcomes that a server can return to a client. These outcomes are intended to be stable across deployments so that clients can react consistently, even when policy and scoring logic differ between servers.

6.1. Overview

A SOUTH server evaluates a request and returns exactly one primary decision. The decision describes how the client should treat the requested operation. The server may include additional metadata, such as a numerical score or a probability value, but the primary decision is authoritative.

This document groups decisions into four categories:

* Allow decisions, where the server authorizes the request.

* Deny decisions, where the server does not authorize the request.

* Retry and defer decisions, where the server indicates that the request might succeed later.

* Error outcomes, where the server cannot evaluate the request.

6.2. Deterministic Allow and Deny

In the deterministic mode, the server returns an explicit allow or deny decision. Clients that do not use stochastic behavior can treat SOUTH as a standard binary authorization protocol.

allow The server authorizes the requested operation. The client is expected to proceed, subject to any local enforcement or application constraints. An allow decision indicates that the server does not require additional checks for this request.

deny The server does not authorize the requested operation. The client must not perform the operation. A deny decision indicates that the server regards the request as incompatible with current policy and that the decision is stable with respect to the information available.

A server that does not implement stochastic behavior or retry semantics can restrict itself to allow and deny decisions.

6.3. Stochastic Allow

SOUTH supports a stochastic mode where the server produces an allow decision based on a probability derived from its internal preference score. In this mode, the server evaluates the request, computes a score in the interval [0, 1], and maps this score to a probability of authorization.

The exact mapping from score to probability is a matter of local policy. A simple example is a threshold rule where scores above a high threshold always allow, scores below a low threshold always deny, and scores in between allow with a probability that increases with the score.

When a server uses stochastic behavior, the primary decision seen by the client is still allow or deny. SOUTH does not define a separate decision token for stochastic allow. Instead, the server may include an optional field that reports the probability or score that was used to produce the decision.

From the client perspective, a stochastic allow is processed in the same way as a deterministic allow. The distinction is a server side choice that affects how often allow decisions are produced near the decision boundary.

6.4. Retry and Defer

Some authorization outcomes are temporary. A server may be overloaded, may be missing required context, or may need the client to perform an additional step before a decision can be made. SOUTH models these situations through retry and defer outcomes.

retry The server cannot authorize the request at this time, but the client may retry after a delay. The response may include a hint such as a suggested retry interval. Clients must treat retry as a deny for the current attempt.

defer The server requests that the client defer the operation and possibly perform an additional step, such as step up verification or context collection. The exact meaning of defer is deployment specific. Clients must not perform the requested operation until a later allow decision is obtained.

Retry and defer decisions allow clients to distinguish between a stable policy deny and a temporary condition. This distinction is important in systems where repeated failures may signal misconfiguration, attack, or changes in policy.

6.5. Error Semantics

A server may be unable to evaluate a request. Causes include malformed input, lack of required authentication, internal faults, or backend unavailability. In such cases, SOUTH distinguishes between an authorization decision and a protocol or processing error.

When a server returns an error, the response indicates that no authorization decision has been made. The client must apply its own failure policy. For example, a client may choose to fail closed and block the operation, or fail open in development environments where availability is more important.

Implementations should distinguish clearly between deny and error. A deny decision is a valid authorization outcome. An error indicates that the server did not complete the evaluation.

6.6. Idempotence and Side Effects

SOUTH evaluates authorization decisions only. It does not perform the requested operation and does not define any side effects beyond logging and internal accounting. Two identical requests sent to the same server under the same conditions may produce the same decision or, in stochastic mode, may produce different outcomes with known probabilities.

Clients should treat SOUTH calls as idempotent with respect to system state. Any non idempotent behavior, such as consuming one time credentials, is a property of the surrounding system, not of SOUTH itself.

7. Request Object

A SOUTH request is a structured object that describes the operation the client wishes to perform. The request object is transport independent. This document uses JSON examples for clarity, but other encodings are possible.

7.1. Fields

A request object contains the fields listed below. Fields marked as required MUST be present. Fields marked as optional MAY be omitted.

request_id (optional) An opaque identifier chosen by the client. When present, the server SHOULD copy this value into the response. The identifier helps clients correlate responses with requests.

subject (required) An object that describes the actor on whose behalf the request is made. The subject object MUST contain at least an id field and MAY contain a type field and an attributes object.

action (required) An object that describes the operation the client wishes to perform. The action object MUST contain a type field and MAY contain additional fields such as name and parameters.

object (required) An object that describes the target of the operation. The object MUST contain at least an id or type field and MAY contain an attributes object.

context (optional) An object that carries additional information about the environment, such as network location, device posture, rates, or time. The structure of the context object is deployment specific.

signals (optional) An object that holds pre computed scores or hints from other modules, such as anomaly detectors or rate limiters. SOUTH does not define specific signal names.

extensions (optional) An object reserved for deployment specific fields that are not covered by the core model. Extensions MUST NOT change the meaning of the core fields.

7.2. Subject

The subject object describes who or what is acting. A minimal subject object contains only an identifier. Richer deployments may include type and attributes. The following JSON examples are illustrative.

```
{
  "subject": {
    "id": "user-1234"
  }
}
```

Figure 1: Basic subject object

```
{
  "subject": {
    "id": "service-api-gateway",
    "type": "service",
    "attributes": {
      "tenant": "tenant-42",
      "roles": ["gateway", "frontend"],
      "auth_method": "mtls"
    }
  }
}
```

Figure 2: Extended subject object

7.3. Action

The action object describes the requested operation. The type field identifies the general category. The name and parameters fields allow further refinement.

```
{
  "action": {
    "type": "tool.invoke",
    "name": "database_query",
    "parameters": {
      "query": "SELECT * FROM customers WHERE id = ?"
    }
  }
}
```

Figure 3: Example action objects

Deployment profiles SHOULD define allowed action types and their expected parameters. SOUTH itself does not interpret action semantics.

7.4. Object

The object describes the target resource. The identification scheme is deployment specific. The object MAY be a logical resource such as a table, an API endpoint, a queue, a file, a node, or a service.

```
{
  "object": {
    "id": "customers-table",
    "type": "database.table",
    "attributes": {
      "sensitivity": "high",
      "owner": "finance-team"
    }
  }
}
```

Figure 4: Example object objects

7.5. Context and Signals

The context and signals fields allow a client to supply additional information that may be relevant for policy evaluation. The structure of these fields is not fixed by SOUTH. Deployments may define profiles that describe common attributes.

```
{
  "context": {
    "ip": "198.51.100.23",
    "location": "region-1",
    "time": "2025-11-27T10:34:12Z",
    "user_agent": "example-client/1.0"
  },
  "signals": {
    "anomaly_score": 0.41,
    "recent_failures": 2
  }
}
```

Figure 5: Example context and signals

Servers MAY ignore unknown context or signal fields. Clients SHOULD NOT rely on any specific field being honored unless the deployment defines a shared profile.

7.6. Complete Request Example

The following example shows a complete SOUTH request object in JSON form. Line breaks and spacing are for readability.

```
{
  "request_id": "550e8400-e29b-41d4-a716-446655440000",
  "subject": {
    "id": "user-1234",
    "type": "user",
    "attributes": {
      "tenant": "tenant-42",
      "roles": ["reader"]
    }
  },
  "action": {
    "type": "tool.invoke",
    "name": "database_query",
    "parameters": {
      "query": "SELECT * FROM customers WHERE id = ?"
    }
  },
  "object": {
    "id": "customers-table",
    "type": "database.table",
    "attributes": {
      "sensitivity": "high"
    }
  },
  "context": {
    "ip": "198.51.100.23",
    "location": "region-1",
    "time": "2025-11-27T10:34:12Z"
  },
  "signals": {
    "anomaly_score": 0.41
  },
  "extensions": {
    "trace_id": "abc123"
  }
}
```

Figure 6: Complete SOUTH request example

8. Response Object

A SOUTH response is a structured object that carries the server decision and optional metadata. The response object is transport independent. This document uses JSON examples for clarity, but other encodings are possible.

8.1. Fields

A response object contains the fields listed below. Fields marked as required MUST be present. Fields marked as optional MAY be omitted.

request_id (optional) An opaque identifier copied from the request. When present, this value MUST match the request_id in the corresponding request. It allows the client to correlate responses with requests.

decision (required) A string that indicates the primary outcome of the evaluation. Valid values are "allow", "deny", "retry", "defer", and "error".

score (optional) A numerical value in the interval [0, 1] that represents the internal preference score used by the server. This field is intended for diagnostics and logging. Clients MUST NOT rely on a specific interpretation of this value unless agreed in a deployment profile.

probability (optional) A numerical value in the interval [0, 1] that represents the probability used when the server applies stochastic behavior to produce an allow decision. This field MAY be present even when the decision is deny, to indicate the probability threshold that was not met.

reason (optional) A short, human readable string that describes the decision at a coarse level, such as "policy", "rate_limit", or "missing_context". The reason is not intended to expose internal policy details.

details (optional) An object that carries structured diagnostic information. The format of the details object is deployment specific.

retry_after (optional) A hint that indicates when the client MAY retry the request. This field is meaningful when the decision is "retry" or "defer". The value MAY be a number of seconds or a timestamp string, depending on the deployment profile.

error_code (optional) A machine readable string that indicates the type of error when the decision is "error". Example values include "invalid_request", "unauthenticated", and "server_error".

extensions (optional) An object reserved for deployment specific fields that are not covered by the core model. Extensions MUST NOT change the meaning of the decision field.

8.2. Allow Responses

When the server authorizes the operation, it returns a response with decision set to "allow". The server MAY include score and probability fields to expose internal evaluation metrics.

```
{
  "request_id": "550e8400-e29b-41d4-a716-446655440000",
  "decision": "allow",
  "reason": "policy",
  "score": 0.98
}
```

Figure 7: Deterministic allow response

```
{
  "request_id": "550e8400-e29b-41d4-a716-446655440000",
  "decision": "allow",
  "reason": "boundary_region",
  "score": 0.62,
  "probability": 0.55
}
```

Figure 8: Stochastic allow response

From the client perspective, a stochastic allow is treated in the same way as a deterministic allow. The probability field is optional metadata that may be logged or used for audit.

8.3. Deny Responses

When the server does not authorize the operation, and the decision is considered stable with respect to current policy and context, it returns decision "deny".

```
{
  "request_id": "550e8400-e29b-41d4-a716-446655440000",
  "decision": "deny",
  "reason": "policy",
  "score": 0.12
}
```

Figure 9: Deny response

Clients MUST NOT perform the requested operation after a deny decision. The server MAY still include score and probability fields for diagnostic purposes, but these values do not change the meaning of the deny outcome.

8.4. Retry and Defer Responses

When the server cannot authorize the request at this time, but the outcome may change later, it can return a decision of "retry" or "defer".

```
{
  "request_id": "550e8400-e29b-41d4-a716-446655440000",
  "decision": "retry",
  "reason": "load",
  "retry_after": 30
}
```

Figure 10: Retry response with hint

```
{
  "request_id": "550e8400-e29b-41d4-a716-446655440000",
  "decision": "defer",
  "reason": "missing_context",
  "details": {
    "required_factor": "stronger_auth"
  }
}
```

Figure 11: Defer response with step up hint

Clients MUST treat retry and defer as not authorized for the current attempt. The client MAY retry later or perform the indicated step, depending on local logic and deployment guidance.

8.5. Error Responses

When the server cannot process the request, it returns a decision of "error". In this case no authorization decision has been made. The client MUST apply its own failure handling policy.

```
{
  "request_id": "550e8400-e29b-41d4-a716-446655440000",
  "decision": "error",
  "error_code": "invalid_request",
  "reason": "malformed_subject"
}
```

Figure 12: Error response

Deployments SHOULD document how clients are expected to behave on error responses. For example, some environments may require fail closed behavior, while others may accept fail open in development or testing stages.

8.6. Correlation and Logging

Implementations SHOULD log request and response pairs for audit, debugging, and incident analysis. The request_id field enables simple correlation. Deployments may also use external correlation identifiers conveyed in the extensions field.

SOUTH does not define log formats or retention policies. Operators are responsible for applying local security and privacy requirements to any stored decision data.

9. Wire Protocol

SOUTH defines a logical exchange between a client and a server: the client sends a request object and receives a response object. The protocol is transport independent. This document specifies a concrete HTTP binding that uses JSON as the serialization format.

9.1. Transport Independent Model

In the abstract model, a client constructs a request object as described in Section 7, sends it to a server, and receives a response object as described in Section Section 8. The binding between these objects and any underlying transport is left to deployment profiles.

The following properties are expected of any SOUTH transport binding.

* The binding MUST preserve the request object and response object without modification, except for encoding.
* The binding MUST provide a way to carry authentication and integrity information for the client and server.
* The binding SHOULD support request correlation, for example by propagating the request_id.
* The binding SHOULD allow clients to distinguish between authorization decisions and transport or protocol errors.

9.2. HTTP Binding

This section specifies an HTTP binding for SOUTH following HTTP semantics [RFC9110]. Other bindings may be defined in separate documents. The HTTP binding uses JSON encoding for request and response bodies.

Clients send SOUTH requests as HTTP POST messages to a server controlled endpoint. The choice of path is a deployment concern. A common pattern is a path such as /south/decision.

A typical HTTP request has the following form.

```
POST /south/decision HTTP/1.1
Host: auth.example.com
Content-Type: application/json
Accept: application/json
Authorization: Bearer eyJhbGciOi...

{
  "request_id": "550e8400-e29b-41d4-a716-446655440000",
  "subject": {
    "id": "user-1234",
    "type": "user"
  },
  "action": {
    "type": "tool.invoke",
    "name": "database_query"
  },
  "object": {
    "id": "customers-table",
    "type": "database.table"
  }
}
```

Figure 13: Example HTTP request

The server responds with a JSON encoded SOUTH response object.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "request_id": "550e8400-e29b-41d4-a716-446655440000",
  "decision": "allow",
  "reason": "policy",
  "score": 0.94
}
```

Figure 14: Example HTTP response

Clients and servers MAY use HTTP authentication mechanisms or carry credentials in headers, such as OAuth bearer tokens or DPoP proofs. SOUTH does not define any specific authentication scheme.

9.3. HTTP Status Codes

In the HTTP binding, the HTTP status code and the SOUTH decision field carry different information. The HTTP status code reflects the result of transport and parsing. The decision field reflects the authorization outcome.

Servers using the HTTP binding SHOULD follow these guidelines.

200 OK The request was successfully processed and a SOUTH response object is present in the body. The decision field indicates the authorization outcome, such as allow, deny, retry, defer, or error.

400 Bad Request The HTTP message could not be parsed or the body could not be decoded as a SOUTH request object. In this case, the server MAY return an error response body, but clients cannot rely on its presence.

401 Unauthorized / 403 Forbidden The client failed authentication or is not permitted to call the SOUTH endpoint. The server MAY omit a SOUTH response object in this case.

500 Internal Server Error and 5xx The server encountered an internal error. A SOUTH response object MAY be returned, but clients MUST treat the condition as a transport or processing failure, not as an explicit authorization decision.

When the server returns 200 OK, clients SHOULD expect a valid SOUTH response object and MUST interpret the decision field as authoritative. When the server returns non 2xx status codes, clients MUST apply local failure handling policies.

9.4. Timeouts and Retries

Timeouts and connection failures are transport level events, not SOUTH decisions. When a client does not receive a response within its configured timeout, the client MUST treat the attempt as if no decision was obtained.

Clients MAY retry the SOUTH request, subject to local rate and backoff policies. Retry behavior on transport failure is separate from retry behavior signaled by a SOUTH decision with value "retry". A server generated retry decision is explicit in the response body and may include a retry_after hint.

Deployments SHOULD document how clients distinguish between these cases and how many retries are permitted before treating the failure as persistent.

9.5. Idempotence and Caching

SOUTH requests are intended to be idempotent with respect to server policy state. Sending the same request multiple times SHOULD NOT by itself change the policy decision, although stochastic behavior may cause different outcomes near the decision boundary.

Clients and intermediaries MAY cache SOUTH responses when allowed by deployment policy. Caching policies are out of scope for this document. Any caching mechanism MUST ensure that sensitive data in responses is handled according to local security and privacy requirements.

10. Operational Modes

SOUTH is intended as a small decision protocol that can be reused across different environments. This section describes several operational modes. These examples are illustrative and do not constrain implementations.

10.1. Agent and Tool Authorization

In agent based deployments, a language model or agent runtime orchestrates tool calls, API invocations, or system level actions on behalf of a user. Before executing a tool call, the agent runtime can call a SOUTH server to obtain an explicit authorization decision for the proposed action.

A typical flow is:

1. The agent plans to call a tool or service, such as a database, filesystem, or external API.

2. The agent runtime constructs a SOUTH request that encodes the user identity as the subject, the tool invocation as the action, and the target resource as the object. The context may include agent identifiers, chain identifiers, and environment details.

3. The SOUTH server evaluates the request using local policy, recent behavior, and any connected signals, and returns a decision.

4. The agent runtime executes the tool call only if the decision is allow. For deny, retry, or defer, the runtime may adjust the plan, ask for user confirmation, or fail the step according to local rules.

This mode separates alignment of model outputs from alignment of actions executed in the environment. The agent may generate many candidate plans, but only those approved by SOUTH are executed.

10.2. Service to Service and Mesh Gateways

In microservice and service mesh environments [SERVICE-MESH], services call each other over RPC or HTTP. Gateways and sidecars often enforce authorization policies for incoming and outgoing calls. SOUTH can be used as an external decision service in this setting.

A sidecar or gateway can:

1. Extract subject information from mutual TLS, tokens, or local identities.

2. Map the requested method and path to an action and object.

3. Construct a SOUTH request with optional context such as zone, workload identity, or rate information.

4. Call the SOUTH server and apply the returned decision to allow or block the upstream call.

This mode centralizes authorization policy while keeping enforcement at gateways and sidecars.

10.3. Mail Transfer Agents

Mail transfer agents (MTAs) often need to decide whether to accept, relay, or reject a message based on sender identity, recipient, content classification, and reputation signals. SOUTH can be used as an authorization step between MTAs.

In one deployment model:

1. When an incoming SMTP transaction reaches a receiving MTA, the MTA builds a SOUTH request capturing envelope sender, envelope recipient, client IP, authentication status, and local reputation scores as signals.

2. The MTA calls a SOUTH server and receives a decision such as allow, deny, or defer.

3. The MTA maps allow to message acceptance, deny to rejection, and defer to temporary failure codes such as 4xx.

This mode lets operators encode complex spam and abuse policies outside the MTA core logic while keeping SMTP behavior standard.

10.4. SDN Controllers and NFV Service Chaining

Software defined networking (SDN) controllers and network function virtualization (NFV) platforms often compute flow rules and service chains in a control plane, then program enforcement points such as switches or virtual functions. SOUTH can be used as an external decision function for these control plane actions.

A controller can:

1. Construct a SOUTH request where the subject is the control application or tenant, the action is a flow or service chain installation, and the object is the target path or chain.

2. Include context such as current load, link health, and security posture.

3. Call the SOUTH server to decide whether the proposed flow or chain is allowed.

4. Install the flow rules only when the decision is allow. For deny, retry, or defer, the controller may choose alternate paths or delay provisioning.

In this mode, SOUTH does not operate on individual packets. It governs control plane actions that change forwarding behavior.

10.5. Kubernetes Admission and Scheduling

In Kubernetes and similar orchestrators, controllers admit and schedule workloads based on policies and cluster state. SOUTH can be used as an external decision point in admission [K8S-ADMISSION] webhooks or scheduler [K8S-SCHEDULER] extensions.

Possible patterns include:

* An admission webhook builds a SOUTH request for each incoming object, such as a Pod or Job, with the subject as the user or service account, the action as create or update, and the object as the resource. A deny decision maps to a failed admission.

* A scheduler plugin calls SOUTH before binding a Pod to a node, using context such as node labels, taints, and current load as part of the request. A deny or defer decision can cause the scheduler to consider different nodes or queue the Pod.

This mode allows operators to centralize certain policies that govern placement and resource access, while leaving Kubernetes semantics intact.

10.6. Library and Local Mode

While SOUTH is defined as a wire protocol, some deployments may embed the decision logic as a library within an application or gateway. In these cases, the application constructs the same request and response objects but calls a local function instead of sending messages over the network.

Library mode can be useful for low latency paths or for environments that later plan to move to a remote SOUTH server. The wire format defined in this document still provides value by giving a stable schema for testing, logging, and future interoperability.

11. Security Considerations

SOUTH is an authorization decision protocol. It does not replace transport security, authentication, or application specific checks. Incorrect deployment or interpretation can weaken system security. This section highlights key considerations.

11.1. Authentication and Integrity

SOUTH assumes that both client and server identities are established by the surrounding transport and authentication mechanisms. The protocol does not define its own authentication scheme.

Deployments SHOULD:

* Use a secure transport such as TLS to protect SOUTH requests and responses from eavesdropping and tampering.

* Authenticate clients and servers using appropriate mechanisms, such as mutual TLS, bearer tokens, or DPoP proofs, depending on the environment.

* Bind subject information in the request to authenticated identities, so that a client cannot claim a different subject than the one established at the transport layer.

If authentication or integrity is not enforced, an attacker may inject SOUTH requests, spoof subjects, or alter response decisions in transit.

11.2. Fail Open and Fail Closed Behavior

SOUTH separates authorization decisions from transport and processing errors. Clients must decide how to behave when no valid SOUTH response is available.

In security sensitive environments, clients SHOULD fail closed when:

* No response is received within a timeout.

* The HTTP status code is not 2xx.

* The response body cannot be parsed as a SOUTH response object.

Failing open in these situations can allow unauthorized actions when the SOUTH server is unavailable or under attack. Any decision to fail open SHOULD be limited to controlled contexts such as development or explicit emergency recovery procedures.

11.3. Stochastic Decisions and Probing

SOUTH allows stochastic behavior near the decision boundary. While this can reduce predictability, it also introduces considerations for probing and analysis by an attacker.

An adversary who can send many similar requests and observe allow or deny outcomes may estimate the underlying score function or probability mapping. This can reveal where decision boundaries lie and may help the attacker craft more effective requests.

Deployments SHOULD:

* Rate limit requests that reach the SOUTH server, especially near sensitive resources.

* Monitor for repeated probing patterns and adjust policy or block abusive clients.

* Consider limiting the precision of exported scores and probabilities, or omitting them entirely in high risk environments.

11.4. Logging and Policy Leakage

SOUTH requests and responses can contain sensitive information, including subject identifiers, resource names, and diagnostic reasons. Logs that capture this data may reveal internal structure, policy rules, or usage patterns to unauthorized parties.

Operators SHOULD:

* Apply access control and retention policies to SOUTH logs.

* Avoid exposing detailed decision reasons or raw request objects to low privilege users or external systems.

* Consider redacting or aggregating fields that might reveal sensitive policy details or user behavior.

In agent deployments, prompts and tool parameters may appear in action or object fields. These can be sensitive and require the same care as application logs that contain prompts or user inputs.

11.5. Dependency and Bypass Risks

SOUTH is only effective when its decisions are enforced. If some code paths or tools bypass SOUTH, then policy enforcement becomes incomplete.

Deployments SHOULD:

* Identify which actions and tools must be guarded by SOUTH and ensure that all corresponding paths call the SOUTH server or library consistently.

* Ensure that enforcement points, such as gateways, sidecars, or agent runtimes, cannot be bypassed by direct access to back end systems.

* Protect configuration that defines which actions require SOUTH decisions, to prevent downgrades.

If attackers can reach underlying resources without going through SOUTH controlled enforcement points, the protocol provides no protection for those paths.

11.6. Denial of Service

SOUTH servers can become a central dependency. If a SOUTH server is unavailable or overloaded, clients may be unable to obtain decisions and may block or fail open, depending on configuration.

An attacker may try to exhaust SOUTH server capacity by sending a large number of requests. To mitigate this risk, deployments SHOULD:

* Apply rate limiting and admission controls to SOUTH endpoints.
* Use resource isolation and scaling mechanisms appropriate for critical infrastructure components.
* Define clear client behavior when SOUTH is unavailable, with a bias toward fail closed in sensitive environments.

11.7. Caching and Replay

Caching SOUTH responses can reduce load and latency, but it introduces risks if cached decisions are reused beyond their valid context.

Deployments that use caching SHOULD:

* Scope caches to specific subjects, objects, and actions, so that decisions are not applied across unrelated contexts.
* Use appropriate lifetimes for cached entries, especially when policy or context can change quickly.
* Protect any cache keys or stored responses against tampering and unauthorized access.

Attackers who can replay old responses in place of fresh decisions may bypass updated policies. Binding cached entries to request attributes and transport layer security is recommended.

11.8. Agent Specific Considerations

When SOUTH is used in agent frameworks, an agent runtime may generate many candidate actions in response to a single user prompt. SOUTH decisions influence which of these actions are executed in the environment.

Operators SHOULD:

* Ensure that all environment changing actions, such as writes, deployments, and external calls, are covered by SOUTH or equivalent controls.
* Treat user prompts and agent generated plans as untrusted inputs and rely on SOUTH decisions to filter dangerous actions.
* Consider additional rate and scope constraints for agents that can generate many tool calls in a short time.

SOUTH does not guarantee that an agent will choose the best or safest plan. It only decides whether a particular requested action is within policy. Other safety layers remain necessary.

12. Privacy Considerations

SOUTH requests and responses can contain personal data and other sensitive information. This section outlines privacy related considerations. It does not replace local legal or regulatory obligations.

12.1. Data Types in SOUTH Messages

A SOUTH request may carry subject identifiers, tenant identifiers, resource identifiers, locations, timestamps, device attributes, anomaly scores, and other contextual signals. In agent deployments, action and object fields may also contain parts of prompts, tool parameters, or derived user data.

A SOUTH response may contain reasons, error codes, and other diagnostics that reveal aspects of policy or user behavior.

Operators SHOULD treat SOUTH messages and logs as sensitive data. They can reveal:

* Which users access which resources, at what times, and from which locations.

* The structure of internal resources such as tables, endpoints, or tools.

* Behavioral patterns, including anomaly scores and recent failures.

12.2. Data Minimization

SOUTH defines a flexible schema for requests. Deployments SHOULD apply data minimization principles when populating fields.

In particular, clients SHOULD:

* Avoid sending full prompts or large payloads in the action or object fields when a compact identifier would suffice.

* Restrict subject attributes to those needed for policy evaluation, rather than copying entire user records.

* Limit context and signal fields to information that has clear value for the decision.

Servers SHOULD discourage deployments that place unnecessary personal data into SOUTH messages and SHOULD document which fields are essential for common policies.

12.3. Retention and Access Control for Logs

SOUTH implementations often log requests and responses for debugging, incident analysis, and compliance. Such logs can form a detailed record of user and agent behavior.

Operators SHOULD:

* Apply access controls to logs so that only authorized personnel and systems can view them.

* Define and enforce retention periods that are consistent with local policy and regulation.

* Consider pseudonymization or anonymization of identifiers when full fidelity is not needed.

* Avoid exporting raw SOUTH logs to third parties unless appropriate agreements and safeguards are in place.

12.4. Agent Context and Prompt Data

When SOUTH is used with agents, the boundary between policy data and user content can be blurred. Tools may carry pieces of prompts, documents, or messages in their parameters. If these are copied into SOUTH requests, the authorization layer may receive more user data than necessary.

To reduce risk, agent runtimes SHOULD:

* Represent tools and resources using stable identifiers where possible, rather than embedding full content.

* Keep free form user text out of SOUTH messages unless the policy explicitly depends on it.

* Treat any prompt fragments or document snippets that do enter SOUTH requests as sensitive and protect them in storage and transit.

Policy authors SHOULD design rules that depend on coarse attributes or classifications rather than raw user content when possible.

12.5. Metadata and Inference Risks

Even when raw content is not present, metadata can support inference about users and organizations. For example, a sequence of SOUTH requests may reveal:

* Work schedules and time zones of specific users.

* Which internal projects or datasets are active.

* Which resources are considered sensitive or high value.

Operators SHOULD consider these inference risks when deciding where SOUTH messages and logs are stored and who can access them. In some cases, aggregating or sampling logs may be preferable to storing full detail.

12.6. Cross Border Transfers and Third Parties

In some deployments, SOUTH servers may run in different regions or under different administrative domains than the calling services. This can create cross border data flows.

Operators SHOULD:

* Verify that sending SOUTH requests to remote regions or third party providers is compatible with local data protection rules.

* Limit which fields are sent across boundaries, and prefer identifiers over raw content when possible.

* Ensure that contracts or agreements with third parties cover the handling of SOUTH messages and logs.

12.7. User Rights and Transparency

In some jurisdictions, users have rights to access, correct, or delete personal data. SOUTH by itself does not define any mechanisms to support these rights.

Deployments that use SOUTH in systems with user facing obligations SHOULD:

* Treat SOUTH logs and configuration as part of the overall data inventory.

* Ensure that user facing documentation describes how authorization and decision logs are used.

* Provide procedures to locate and handle personal data that appears in SOUTH messages when required by law or policy.

12.8. Extensions and Profiles

The extensions field in SOUTH requests and responses allows deployments to add custom data. Unconstrained use of extensions can introduce privacy risks.

Profiles that define extensions SHOULD:

* Document the purpose and data types of each extension clearly.

* Avoid placing raw user content into extensions when derived attributes would suffice.

* Apply the same security and privacy controls to extension fields as to core fields.

Operators SHOULD periodically review extensions and remove those that are no longer necessary.

13. IANA Considerations

This document registers a new media type for SOUTH request and response objects encoded in JSON. The registration follows the guidelines in [RFC6838].

13.1. application/south+json Media Type

The following media type registration is for the JSON encoding of SOUTH messages.

Type name: application

Subtype name: south+json

Required parameters: None

Optional parameters: None

Encoding considerations: Must be encoded as UTF-8. See [RFC8259].

Security considerations: See the Security Considerations section of this document.

Interoperability considerations: None known.

Published specification: This document.

Applications that use this media type: SOUTH-aware clients and servers, agent frameworks, service meshes, authorization middleboxes, and schedulers requiring structured authorization decisions.

Additional information: None.

Person & email address to contact for further information: Madhava Gaikwad <gaikwad.madhav@gmail.com>

Intended usage: Common

Restrictions on usage: None

Author: See Author section of this document.

Change controller: IESG <iesg@ietf.org>, or as designated by future

standardization work.

14. Normative References

[RFC6838] Freed, N., "Media Type Specifications and Registration Procedures", January 2013, <https://www.rfc-editor.org/rfc/rfc6838>.

[RFC8259] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC9110] "HTTP Semantics", RFC 9110, <https://www.rfc-editor.org/rfc/rfc9110>.

[RFC8446] "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC6749] "The OAuth 2.0 Authorization Framework", RFC 6749, <https://www.rfc-editor.org/rfc/rfc6749>.

[RFC9449] "Demonstrating Proof-of-Possession at the Application Layer (DPoP)", RFC 9449, <https://www.rfc-editor.org/rfc/rfc9449>.

15. Informative References

[NIST-ZT] "Zero Trust Architecture", NIST Special Publication 800-207, <https://csrc.nist.gov/publications/detail/sp/800-207/final>.

[K8S-ADMISSION] "Kubernetes Admission Controllers", Kubernetes Documentation Admission Controllers, <https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/>.

[K8S-SCHEDULER] "Kubernetes Scheduler", Kubernetes Documentation Scheduler, <https://kubernetes.io/docs/concepts/scheduling-eviction/kube-scheduler/>.

[SERVICE-MESH] "Istio Service Mesh Documentation", Istio Documentation Overview, <https://istio.io/latest/docs/>.

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

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

Appendix A. Example Flows

This appendix provides example SOUTH message sequences for several deployment scenarios. These examples are illustrative. They do not constrain implementations.

A.1. Agent Tool Invocation

In this flow a language model based agent wishes to call a database tool on behalf of a user. The agent runtime consults SOUTH before making the tool call.

Step 1: the user submits a prompt to the agent. The agent decides that it needs to read a customer record and plans a tool invocation such as database_query.

Step 2: the agent runtime constructs a SOUTH request. The subject is the user, the action is the tool invocation, the object is the target table, and the context includes agent related information.

```
{
  "request_id": "req-001",
  "subject": {
    "id": "user-1234",
    "type": "user",
    "attributes": {
      "tenant": "tenant-42",
      "roles": ["reader"]
    }
  },
  "action": {
    "type": "tool.invoke",
    "name": "database_query",
    "parameters": {
      "operation": "select",
      "resource": "customers-table"
    }
  },
  "object": {
    "id": "customers-table",
    "type": "database.table"
  },
  "context": {
    "agent_id": "agent-runtime-1",
    "chain_id": "plan-abc",
    "ip": "198.51.100.23"
  }
}
```

Figure 15: Agent tool invocation SOUTH request

Step 3: the SOUTH server evaluates the request and returns a response that allows the action.

```
{
  "request_id": "req-001",
  "decision": "allow",
  "reason": "policy",
  "score": 0.92
}
```

Figure 16: Agent tool invocation SOUTH response

Step 4: the agent runtime proceeds with the database tool call. If the decision had been deny or retry, the runtime would select a different plan or ask the user for guidance.

A.2. Service Mesh Gateway

In this flow a service mesh sidecar consults SOUTH before forwarding an HTTP call from service A to service B.

Step 1: service A sends an HTTP request to service B. The request passes through a sidecar proxy.

Step 2: the sidecar constructs a SOUTH request. The subject is the identity of service A, the action encodes the HTTP method, and the object identifies the upstream service and path.

```
{
  "request_id": "mesh-123",
  "subject": {
    "id": "service-A",
    "type": "service",
    "attributes": {
      "namespace": "payments"
    }
  },
  "action": {
    "type": "http.call",
    "name": "GET",
    "parameters": {
      "path": "/v1/charges"
    }
  },
  "object": {
    "id": "service-B",
    "type": "service",
    "attributes": {
      "namespace": "billing"
    }
  },
  "context": {
    "ip": "10.0.0.5",
    "zone": "zone-1"
  }
}
```

Figure 17: Service mesh SOUTH request

Step 3: the SOUTH server returns a deny decision. The sidecar returns an HTTP error to service A and does not forward the request to service B.

```
{
  "request_id": "mesh-123",
  "decision": "deny",
  "reason": "policy",
  "score": 0.08
}
```

Figure 18: Service mesh SOUTH response

A.3. Mail Transfer Agent

In this flow an incoming SMTP connection triggers a SOUTH check before message acceptance.

Step 1: a sending MTA connects to a receiving MTA and issues an SMTP transaction with MAIL FROM and RCPT TO commands.

Step 2: before accepting the message body, the receiving MTA constructs a SOUTH request. The subject reflects the sending host and authenticated identity, the object is the recipient, and the signals include local reputation scores.

```
{
  "request_id": "mta-42",
  "subject": {
    "id": "smtp-client-1",
    "type": "mta",
    "attributes": {
      "ip": "203.0.113.5",
      "helo": "mail.example.org"
    }
  },
  "action": {
    "type": "smtp.accept",
    "name": "deliver"
  },
  "object": {
    "id": "user@example.net",
    "type": "mailbox"
  },
  "signals": {
    "reputation_score": 0.21,
    "spf_pass": false
  }
}
```

Figure 19: MTA SOUTH request

Step 3: the SOUTH server returns a defer decision. The MTA responds with a temporary failure code and may try again later or drop the connection according to local policy.

Appendix B. Signing Profiles

SOUTH does not define its own authentication or signing format. This appendix sketches profiles that combine SOUTH with existing mechanisms. These descriptions are informative.

B.1. Mutual TLS Profile

In the mutual TLS profile, client and server authentication rely on TLS [RFC8446]. The client presents a certificate and the server maps the certificate to a client identity. The SOUTH request includes a subject that is derived from this identity.

Properties:

* Integrity and confidentiality are provided by the TLS session.
* The subject in the SOUTH request is bound to the certificate presented on the connection.
* No additional signatures are required in the body.

This profile is suitable inside data centers or service mesh environments where mutual TLS is already deployed.

B.2. OAuth 2.0 and DPoP Profile

In this profile, the client authenticates to the SOUTH server using OAuth 2.0 bearer tokens [RFC6749] and may use DPoP [RFC9449] for proof of possession.

A typical flow is:

1. The client obtains an access token from an authorization server, with a subject identifier and optional scopes.
2. The client sends SOUTH requests over HTTPS, including the token in an Authorization header and optional DPoP proof.
3. The SOUTH server validates the token and proof, then uses the subject and scopes as part of policy evaluation.

The SOUTH subject should reflect the identity in the access token. Additional attributes may be derived from token claims.

B.3. Signed Request Bodies

Some deployments may require explicit integrity protection for the SOUTH request body, separate from the transport. One option is to use a JSON based signature format that covers the serialized request object.

A simple pattern is:

1. Serialize the SOUTH request object as JSON.
2. Compute a signature over the JSON representation using a key associated with the client.
3. Attach the signature and key identifier in the extensions field or a separate header that is understood by the SOUTH server.

The server verifies the signature before evaluating policy. The exact signature format, key management, and canonical serialization rules are deployment specific and out of scope for this document.

B.4. Agent and User Binding

In agent scenarios, a SOUTH request may represent a combined decision for a user and an agent runtime. One deployment model uses two layers of proof.

* The outer transport authenticates the agent runtime itself, for example through mutual TLS or a client credential token.
* The inner subject reflects the user identity, which is carried in a user token or session credential.

The SOUTH server can validate both identities and apply policy that depends on the combination. For example, some tools may only be allowed when a specific agent acts on behalf of a given user group.

Appendix C. Rate and Load Aware Policies

SOUTH is compatible with rate and load sensitive decision logic. This appendix outlines patterns for encoding such policies in the request and response objects.

C.1. Rate and Load Signals

Rate and load information can appear in the context and signals fields of the SOUTH request. The following fields are common examples.

* Per subject request rate over a recent window.
* Per resource or per tenant request rate.
* Current load on a target service or cluster.
* Anomaly or burst scores from external detectors.

The exact schema for these fields is deployment specific. A profile can define standard names such as subject_rate or service_load.

C.2. Decisions for Throttling and Shedding

Rate and load aware policies can use SOUTH decisions to control throttling and shedding.

Typical patterns include:

* allow when rates and load are below thresholds.
* deny when a request clearly exceeds policy or is part of an abusive pattern.
* retry when the policy suggests that the operation may succeed later, for example under transient load.
* defer when the system prefers queued or delayed execution rather than immediate failure.

The reason field and optional retry_after extension can give further guidance to clients. For example, a rate limited decision may suggest a backoff period.

C.3. Agent Specific Rate Policies

Agents can generate many actions in a short time. SOUTH can help enforce limits that prevent unbounded tool use or external calls.

Examples include:

* Per user and per agent caps on tool calls per minute.

* Budgets for external API calls or high cost operations.

* Step up requirements when an agent exceeds typical behavior, such as additional user confirmation.

These policies are expressed as functions of the rate related signals in the SOUTH request. The SOUTH response then guides how the agent runtime adapts its plan.

C.4. Fairness and Multi Tenant Settings

In multi tenant environments, rate and load policies affect fairness. SOUTH can help implement per tenant and per group quotas that prevent a single tenant from consuming a disproportionate share of capacity.

A SOUTH server can:

1. Track usage counters per tenant or group.

2. Include current quota status in the decision logic.

3. Return deny or defer decisions when quotas are exceeded.

The protocol itself does not enforce quotas, but it provides a structured place to apply quota aware decisions and to signal these decisions back to clients.

Author’s Address

Madhava Gaikwad
Independent Researcher
Email: gaikwad.madhav@gmail.com