API Reference

This guide provides practical examples for integrating with the ATB APIs. For complete specifications, see the OpenAPI docs: - Broker API - Main enforcement gateway - AgentAuth API - Authorization service

---

Quick Reference

Endpoint	Method	Service	Purpose
`/health`	GET	Both	Health check
`/authorize`	POST	AgentAuth	Request PoA token (low-risk)
`/challenge`	POST	AgentAuth	Create approval challenge
`/challenge/{id}/approve`	POST	AgentAuth	Submit approval
`/challenge/{id}/complete`	POST	AgentAuth	Get PoA token after approval
`/*`	ANY	Broker	Proxy to upstream with PoA validation

Authentication

All requests require mTLS with SPIFFE certificates.

Headers

Header	Required	Description
`X-Poa-Token`	Yes (Broker)	Signed PoA JWT token
`X-Request-Id`	No	Correlation ID for tracing
`Content-Type`	Yes (POST)	`application/json`

AgentAuth API

Health Check

```
curl -k https://localhost:8444/health
```

Response:

```
{"status": "healthy", "version": "1.0.0"}
```

Request PoA Token (Low-Risk)

For low-risk actions that don't require approval.

```
curl -k --cert client.crt --key client.key \
  -X POST https://localhost:8444/authorize \
  -H "Content-Type: application/json" \
  -d '{
    "action": "system.status.read",
    "constraints": {},
    "legal_basis": {
      "type": "legitimate_interest",
      "accountable_party": "ops-team@example.com"
    }
  }'
```

Response:

```
{
  "token": "eyJhbGciOiJFZERTQSIsInR5cCI6IkpXVCJ9...",
  "expires_at": "2026-01-12T10:25:00Z",
  "action": "system.status.read",
  "risk_tier": "low"
}
```

Create Approval Challenge (Medium/High-Risk)

For actions requiring human approval.

```
curl -k --cert client.crt --key client.key \
  -X POST https://localhost:8444/challenge \
  -H "Content-Type: application/json" \
  -d '{
    "action": "sap.vendor.change",
    "constraints": {
      "vendor_id": "V-12345",
      "changes": ["bank_account"]
    },
    "legal_basis": {
      "type": "contract",
      "accountable_party": "finance@example.com"
    },
    "context": {
      "reason": "Update vendor payment details per contract amendment",
      "ticket": "JIRA-1234"
    }
  }'
```

Response:

```
{
  "challenge_id": "ch_abc123def456",
  "action": "sap.vendor.change",
  "risk_tier": "high",
  "required_approvers": 2,
  "current_approvers": 0,
  "expires_at": "2026-01-12T10:25:00Z",
  "approval_url": "https://approvals.example.com/ch_abc123def456"
}
```

Submit Approval

Approvers submit their approval using this endpoint.

```
curl -k --cert approver.crt --key approver.key \
  -X POST https://localhost:8444/challenge/ch_abc123def456/approve \
  -H "Content-Type: application/json" \
  -d '{
    "approver_id": "alice@example.com",
    "decision": "approve",
    "comment": "Verified against contract amendment #42"
  }'
```

Response:

```
{
  "challenge_id": "ch_abc123def456",
  "status": "pending",
  "current_approvers": 1,
  "required_approvers": 2,
  "approvals": [
    {
      "approver_id": "alice@example.com",
      "decision": "approve",
      "timestamp": "2026-01-12T10:21:00Z"
    }
  ]
}
```

Complete Challenge (Get PoA Token)

After sufficient approvals, retrieve the PoA token.

```
curl -k --cert client.crt --key client.key \
  -X POST https://localhost:8444/challenge/ch_abc123def456/complete
```

Response (success):

```
{
  "token": "eyJhbGciOiJFZERTQSIsInR5cCI6IkpXVCJ9...",
  "expires_at": "2026-01-12T10:25:00Z",
  "action": "sap.vendor.change",
  "risk_tier": "high",
  "approvers": ["alice@example.com", "bob@example.com"]
}
```

Response (insufficient approvals):

```
{
  "error": "insufficient_approvals",
  "message": "Requires 2 approvers, currently have 1",
  "challenge_id": "ch_abc123def456",
  "current_approvers": 1,
```

```
  "required_approvers": 2
}
```

Reject Challenge

Approvers can reject a challenge.

```
curl -k --cert approver.crt --key approver.key \
  -X POST https://localhost:8444/challenge/ch_abc123def456/approve \
  -H "Content-Type: application/json" \
  -d '{
    "approver_id": "carol@example.com",
    "decision": "reject",
    "comment": "Vendor details do not match our records"
  }'
```

Response:

```
{
  "challenge_id": "ch_abc123def456",
  "status": "rejected",
  "rejected_by": "carol@example.com",
  "reason": "Vendor details do not match our records"
}
```

Broker API

Health Check

```
curl -k https://localhost:8443/health
```

Response:

```
{"status": "healthy", "version": "1.0.0"}
```

Proxied Request with PoA

Include the PoA token in requests to upstream services.

```
# First, get a PoA token from AgentAuth
POA_TOKEN=$(curl -sk --cert client.crt --key client.key \
  -X POST https://localhost:8444/authorize \
  -H "Content-Type: application/json" \
  -d '{"action": "crm.contact.read", "constraints": {"contact_id": "C-123"}}' \
  | jq -r '.token')

# Then, make the request through the Broker
curl -k --cert client.crt --key client.key \
  -X GET https://localhost:8443/crm/contacts/C-123 \
  -H "X-Poa-Token: $POA_TOKEN"
```

Response (success): The upstream response is returned transparently:

```
{
  "id": "C-123",
  "name": "John Doe",
  "email": "john@example.com"
}
```

Request Without PoA

Requests without a PoA token are denied.

```
curl -k --cert client.crt --key client.key \
  -X GET https://localhost:8443/crm/contacts/C-123
```

Response:

```
{
  "error": "missing_poa",
  "message": "X-Poa-Token header is required"
}
```

HTTP Status: `401 Unauthorized`

Request with Invalid PoA

```
curl -k --cert client.crt --key client.key \
  -X GET https://localhost:8443/crm/contacts/C-123 \
  -H "X-Poa-Token: invalid-token"
```

Response:

```
{
  "error": "invalid_poa_signature",
  "message": "PoA token signature verification failed"
}
```

HTTP Status: `401 Unauthorized`

Request with Expired PoA

```
{
  "error": "token_expired",
  "message": "PoA token expired at 2026-01-12T10:20:00Z"
}
```

HTTP Status: `401 Unauthorized`

Request with Constraint Violation

If the request violates PoA constraints:

```
{
  "error": "constraint_violation",
  "message": "Request amount 50000 exceeds constraint max_amount 10000"
}
```

HTTP Status: `403 Forbidden`

Error Codes

Code	HTTP Status	Description
`missing_poa`	401	No PoA token provided
`invalid_poa_signature`	401	Token signature invalid
`token_expired`	401	Token has expired
`token_not_yet_valid`	401	Token `nbf` is in the future
`token_already_used`	401	Token JTI was already seen
`action_not_allowed`	403	Action not in policy
`constraint_violation`	403	Constraint limits exceeded
`insufficient_approvals`	403	Need more approvers
`challenge_not_found`	404	Challenge ID doesn't exist
`challenge_expired`	410	Challenge has expired
`upstream_error`	502	Upstream returned error
`upstream_timeout`	504	Upstream request timed out

SDK Examples

Python

```python
import requests
import jwt
import time
import uuid

class ATBClient:
    def __init__(self, broker_url, agentauth_url, cert, key, ca):
        self.broker_url = broker_url
        self.agentauth_url = agentauth_url
        self.session = requests.Session()
        self.session.cert = (cert, key)
        self.session.verify = ca

    def authorize(self, action, constraints=None, legal_basis=None):
        """Get PoA token for an action."""
        resp = self.session.post(
            f"{self.agentauth_url}/authorize",
            json={
                "action": action,
                "constraints": constraints or {},
                "legal_basis": legal_basis or {
                    "type": "legitimate_interest",
```

```
                    "accountable_party": "agent@example.com"
                }
            }
        )
        resp.raise_for_status()
        return resp.json()["token"]

    def request(self, method, path, poa_token, **kwargs):
        """Make request through broker with PoA token."""
        headers = kwargs.pop("headers", {})
        headers["X-Poa-Token"] = poa_token

        resp = self.session.request(
            method,
            f"{self.broker_url}{path}",
            headers=headers,
            **kwargs
        )
        return resp

# Usage
client = ATBClient(
    broker_url="https://localhost:8443",
    agentauth_url="https://localhost:8444",
    cert="client.crt",
    key="client.key",
    ca="ca.crt"
)

# Low-risk action
token = client.authorize("system.status.read")
resp = client.request("GET", "/system/status", token)
print(resp.json())
```

Go

```
package main

import (
    "bytes"
    "crypto/tls"
    "crypto/x509"
    "encoding/json"
    "fmt"
    "io"
    "net/http"
    "os"
)

type ATBClient struct {
    BrokerURL    string
    AgentAuthURL string
    HTTPClient   *http.Client
}
```

```go
type AuthorizeRequest struct {
    Action      string                 `json:"action"`
    Constraints map[string]interface{} `json:"constraints,omitempty"`
    LegalBasis  map[string]interface{} `json:"legal_basis,omitempty"`
}

type AuthorizeResponse struct {
    Token     string `json:"token"`
    ExpiresAt string `json:"expires_at"`
    RiskTier  string `json:"risk_tier"`
}

func NewATBClient(brokerURL, agentAuthURL, certFile, keyFile, caFile string) (*ATBClient, erro
    cert, err := tls.LoadX509KeyPair(certFile, keyFile)
    if err != nil {
        return nil, err
    }

    caCert, err := os.ReadFile(caFile)
    if err != nil {
        return nil, err
    }
    caCertPool := x509.NewCertPool()
    caCertPool.AppendCertsFromPEM(caCert)

    tlsConfig := &tls.Config{
        Certificates: []tls.Certificate{cert},
        RootCAs:      caCertPool,
    }

    return &ATBClient{
        BrokerURL:    brokerURL,
        AgentAuthURL: agentAuthURL,
        HTTPClient: &http.Client{
            Transport: &http.Transport{TLSClientConfig: tlsConfig},
        },
    }, nil
}

func (c *ATBClient) Authorize(action string, constraints map[string]interface{}) (string, erro
    req := AuthorizeRequest{
        Action:      action,
        Constraints: constraints,
        LegalBasis: map[string]interface{}{
            "type":              "legitimate_interest",
            "accountable_party": "agent@example.com",
        },
    }

    body, _ := json.Marshal(req)
    resp, err := c.HTTPClient.Post(
        c.AgentAuthURL+"/authorize",
        "application/json",
```

```
        bytes.NewBuffer(body),
    )
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    var authResp AuthorizeResponse
    json.NewDecoder(resp.Body).Decode(&authResp)
    return authResp.Token, nil
}

func (c *ATBClient) Request(method, path, poaToken string) (*http.Response, error) {
    req, _ := http.NewRequest(method, c.BrokerURL+path, nil)
    req.Header.Set("X-Poa-Token", poaToken)
    return c.HTTPClient.Do(req)
}

func main() {
    client, _ := NewATBClient(
        "https://localhost:8443",
        "https://localhost:8444",
        "client.crt",
        "client.key",
        "ca.crt",
    )

    token, _ := client.Authorize("system.status.read", nil)
    resp, _ := client.Request("GET", "/system/status", token)
    defer resp.Body.Close()

    body, _ := io.ReadAll(resp.Body)
    fmt.Println(string(body))
}
```

Rate Limits

Endpoint	Limit	Window
`/authorize`	100	1 minute
`/challenge`	20	1 minute
`/challenge/*/approve`	50	1 minute
`/*` (Broker)	1000	1 minute

Exceeded limits return HTTP `429 Too Many Requests`:

```
{
  "error": "rate_limit_exceeded",
  "message": "Rate limit exceeded, retry after 30 seconds",
  "retry_after": 30
}
```

Webhooks

Configure webhooks for approval notifications:

```yaml
webhooks:
  - url: "https://slack.example.com/webhook"
    events: ["challenge.created", "challenge.approved", "challenge.rejected"]
    secret: "${WEBHOOK_SECRET}"
```

Webhook Payload:

```json
{
  "event": "challenge.created",
  "timestamp": "2026-01-12T10:20:00Z",
  "data": {
    "challenge_id": "ch_abc123def456",
    "action": "sap.vendor.change",
    "risk_tier": "high",
    "requester": "spiffe://example.org/agent/crm",
    "required_approvers": 2,
    "expires_at": "2026-01-12T10:25:00Z",
    "context": {
      "reason": "Update vendor payment details",
      "ticket": "JIRA-1234"
    }
  }
}
```

Related Documentation

- Authentication Guide - Token and identity details
- Getting Started - Quick start tutorial
- SDK Documentation - Client libraries
- Examples - Complete code examples