DEVELOPER TRAINING

Anti-Call Masking Platform - API Integration

Duration: ~45 minutes
Audience: Integration Developers, API Consumers
Version: 2.0 | January 2026

PREREQUISITES

- Completed System Overview training
- API credentials (sandbox environment)
- Familiarity with REST APIs
- Understanding of JSON and HTTP
- Development environment setup

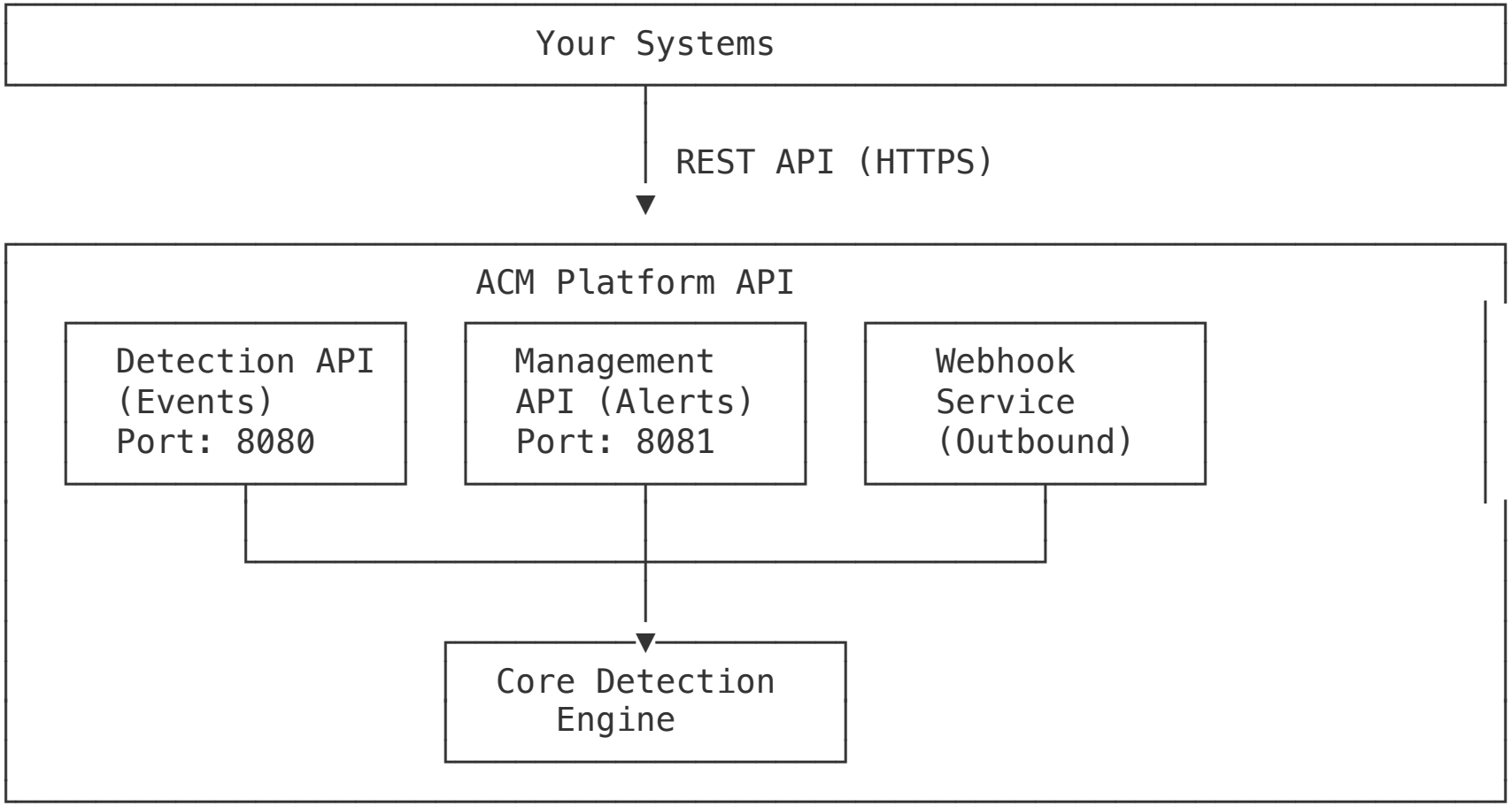
AGENDA

1. API Architecture Overview
2. Authentication & Authorization
3. Core API Endpoints
4. Event Submission
5. Alert Integration
6. Webhooks
7. Error Handling

API ARCHITECTURE OVERVIEW

Understanding the integration landscape

INTEGRATION ARCHITECTURE



API ENDPOINTS SUMMARY

API	Base URL	Purpose
Detection	`/api/v1/events`	Submit call events
Alerts	`/api/v1/alerts`	Query/manage alerts
Config	`/api/v1/config`	System configuration
Webhooks	`/api/v1/webhooks`	Manage subscriptions
Health	`/health`	System status

API VERSIONING

> Current version: **v1**
> Version is included in the URL path

```
GET /api/v1/alerts HTTP/1.1
Host: acm-api.yourcompany.com
```

VERSIONING POLICY:

- Breaking changes = new major version
- Additive changes = same version
- Deprecated endpoints: 6 months notice

RATE LIMITS

Endpoint	Rate Limit	Burst
Event submission	150,000/sec	200,000/sec
Alert queries	1,000/min	2,000/min
Config updates	100/min	200/min
Batch operations	100/min	150/min

Headers: `X-RateLimit-Remaining`, `X-RateLimit-Reset`

AUTHENTICATION & AUTHORIZATION

Securing your integration

AUTHENTICATION METHODS

Method	Use Case	Format
API Key	Server-to-server	`X-API-Key: key`
OAuth 2.0	User-facing apps	`Authorization: Bearer token`
mTLS	High-security	Client certificate

API KEY AUTHENTICATION

```
# Using API Key header
curl -X POST https://acm-api.yourcompany.com/api/v1/events \
  -H "X-API-Key: your-api-key-here" \
  -H "Content-Type: application/json" \
  -d '{"a_number": "+234801111111", "b_number": "+234802222222"}'
```

> **Security:** Never expose API keys in client-side code or version control!

OAUTH 2.0 FLOW

```
# Step 1: Get access token
curl -X POST https://acm-api.yourcompany.com/oauth/token \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "grant_type=client_credentials" \
  -d "client_id=your-client-id" \
  -d "client_secret=your-client-secret" \
  -d "scope=events:write alerts:read"

# Response:
{
  "access_token": "eyJhbGciOiJSUzI1NiIs...",
  "token_type": "Bearer",
  "expires_in": 3600,
  "scope": "events:write alerts:read"
}
```

USING BEARER TOKEN

```
# Step 2: Use access token
curl -X GET https://acm-api.yourcompany.com/api/v1/alerts \
  -H "Authorization: Bearer eyJhbGciOiJSUzI1NiIs..."
```

TOKEN REFRESH:

- Tokens expire in 1 hour
- Request new token before expiry
- Implement automatic refresh in your client

PERMISSION SCOPES

Scope	Permissions
`events:write`	Submit call events
`alerts:read`	Query alerts
`alerts:write`	Update alert status
`config:read`	Read configuration
`config:write`	Modify configuration
`webhooks:manage`	Manage webhook subscriptions

CORE API ENDPOINTS

Essential operations

HEALTH CHECK

```
GET /health HTTP/1.1
Host: acm-api.yourcompany.com
```

```
// Response 200 OK
{
  "status": "healthy",
  "version": "2.0.0",
  "timestamp": "2026-01-15T10:30:00Z",
  "components": {
    "detection": "healthy",
    "cache": "healthy",
    "database": "healthy"
  }
}
```

Use this endpoint for monitoring and readiness probes.

API DISCOVERY

```
GET /api/v1 HTTP/1.1
Host: acm-api.yourcompany.com
```

```
// Response 200 OK
{
  "version": "v1",
  "endpoints": {
    "events": "/api/v1/events",
    "alerts": "/api/v1/alerts",
    "config": "/api/v1/config",
    "webhooks": "/api/v1/webhooks"
  },
  "documentation": "https://docs.acm.yourcompany.com"
}
```

COMMON RESPONSE FORMAT

```
// Successful response
{
  "success": true,
  "data": { ... },
  "meta": {
    "request_id": "req_abc123",
    "timestamp": "2026-01-15T10:30:00Z"
  }
}

// Error response
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid phone number format",
```

EVENT SUBMISSION

Sending call events for detection

SINGLE EVENT SUBMISSION

```
POST /api/v1/events HTTP/1.1
Host: acm-api.yourcompany.com
X-API-Key: your-api-key
Content-Type: application/json

{
  "a_number": "+2348011111111",
  "b_number": "+2348022222222",
  "timestamp": "2026-01-15T10:30:00.123Z",
  "call_id": "call-uuid-12345",
  "trunk_id": "trunk-001",
  "metadata": {
    "source_ip": "10.0.0.1",
    "codec": "G.711"
  }
}
```

EVENT FIELDS

Field	Type	Required	Description
`a_number`	string	Yes	Caller ID (E.164 format)
`b_number`	string	Yes	Destination (E.164 format)
`timestamp`	ISO 8601	No*	Event time (default: now)
`call_id`	string	No	Unique call identifier
`trunk_id`	string	No	Source trunk identifier
`metadata`	object	No	Additional context

EVENT RESPONSE

```
// Response 202 Accepted
{
  "success": true,
  "data": {
    "event_id": "evt_789xyz",
    "processed": true,
    "alert_generated": false
  }
}

// If alert was generated
{
  "success": true,
  "data": {
    "event_id": "evt_789xyz",
    "processed": true,
```

BATCH EVENT SUBMISSION

```
POST /api/v1/events/batch HTTP/1.1
Host: acm-api.yourcompany.com
X-API-Key: your-api-key
Content-Type: application/json

{
  "events": [
    {
      "a_number": "+2348011111111",
      "b_number": "+2348022222222",
      "timestamp": "2026-01-15T10:30:00.123Z"
    },
    {
      "a_number": "+2348033333333",
      "b_number": "+2348022222222",
      "timestamp": "2026-01-15T10:30:00.456Z"
```

> Batch limit: **1000 events** per request

LATENCY CONSIDERATIONS

<1ms
Detection Time

<5ms
API Response

TIPS FOR LOW LATENCY:

- Use persistent HTTP connections
- Deploy close to ACM servers
- Use batch submission for high volume
- Implement async submission if possible

ALERT INTEGRATION

Querying and managing alerts

LIST ALERTS

```
GET /api/v1/alerts?status=new&severity=critical&limit=50 HTTP/1.1
Host: acm-api.yourcompany.com
Authorization: Bearer your-token
```

QUERY PARAMETERS:

Parameter	Type	Description
`status`	string	new, acknowledged, resolved
`severity`	string	critical, high, medium, low
`since`	ISO 8601	Alerts after this time
`limit`	integer	Max results (default: 100)
`offset`	integer	Pagination offset

ALERT RESPONSE

```
{
  "success": true,
  "data": {
    "alerts": [
      {
        "id": "ACM-2026-001234",
        "severity": "critical",
        "status": "new",
        "b_number": "+2348022222222",
        "a_number_count": 7,
        "detection_window_ms": 4200,
        "detected_at": "2026-01-15T10:30:00Z",
        "a_numbers": [
          "+2348011111111",
          "+2348033333333",
          // ... more
        ]
```

GET SINGLE ALERT

```
GET /api/v1/alerts/ACM-2026-001234 HTTP/1.1
Host: acm-api.yourcompany.com
Authorization: Bearer your-token
```

```
{
  "success": true,
  "data": {
    "id": "ACM-2026-001234",
    "severity": "critical",
    "status": "investigating",
    "b_number": "+2348022222222",
    "a_number_count": 7,
    "calls": [
      {
        "a_number": "+2348011111111",
        "timestamp": "2026-01-15T10:30:00.100Z",
        "call_id": "call-001",
        "trunk_id": "trunk-001",
        "disconnected": true
      }
```

UPDATE ALERT STATUS

```
PATCH /api/v1/alerts/ACM-2026-001234 HTTP/1.1
Host: acm-api.yourcompany.com
Authorization: Bearer your-token
Content-Type: application/json

{
  "status": "resolved",
  "resolution": "confirmed_fraud",
  "notes": "Confirmed masking attack from compromised gateway. Pattern blocked.
}
```

RESOLUTION TYPES:

- `confirmed_fraud` - Verified fraud
- `false_positive` - Legitimate traffic
- `inconclusive` - Unable to determine

DISCONNECT CALLS

```
POST /api/v1/alerts/ACM-2026-001234/disconnect HTTP/1.1
Host: acm-api.yourcompany.com
Authorization: Bearer your-token
Content-Type: application/json

{
  "call_ids": ["call-001", "call-002"],  // Optional: specific calls
  "all": false,                           // Or disconnect all
  "reason": "Confirmed fraud attack"
}
```

```
// Response 200 OK
{
  "success": true,
  "data": {
    "disconnected": 2,
    "failed": 0,
    "already_ended": 5
  }
}
```

WEBHOOKS

Real-time event notifications

WEBHOOK ARCHITECTURE

Features:
- Retry with exponential backoff
- Signature verification
- Event filtering
- Delivery tracking

REGISTER WEBHOOK

```
POST /api/v1/webhooks HTTP/1.1
Host: acm-api.yourcompany.com
Authorization: Bearer your-token
Content-Type: application/json

{
  "url": "https://your-system.com/webhook/acm",
  "events": ["alert.created", "alert.updated"],
  "secret": "your-webhook-secret",
  "filters": {
    "min_severity": "high"
  }
}
```

```
// Response 201 Created
{
  "success": true,
  "data": {
    "id": "wh_abc123",
    "url": "https://your-system.com/webhook/acm",
    "events": ["alert.created", "alert.updated"],
```

AVAILABLE EVENTS

Event	Triggered When
`alert.created`	New alert detected
`alert.updated`	Alert status changed
`alert.resolved`	Alert marked resolved
`calls.disconnected`	Calls terminated
`whitelist.updated`	Whitelist changed
`system.health`	Health status change

WEBHOOK PAYLOAD

```
{
  "event": "alert.created",
  "timestamp": "2026-01-15T10:30:00.123Z",
  "webhook_id": "wh_abc123",
  "delivery_id": "del_xyz789",
  "data": {
    "alert": {
      "id": "ACM-2026-001234",
      "severity": "critical",
      "b_number": "+2348022222222",
      "a_number_count": 7,
      "detected_at": "2026-01-15T10:30:00Z"
    }
  }
}
```

SIGNATURE VERIFICATION

```
// Node.js example
const crypto = require('crypto');

function verifyWebhook(payload, signature, secret) {
  const expectedSignature = crypto
    .createHmac('sha256', secret)
    .update(payload, 'utf8')
    .digest('hex');

  return crypto.timingSafeEqual(
    Buffer.from(signature),
    Buffer.from('sha256=' + expectedSignature)
  );
}

// Express handler
```

WEBHOOK BEST PRACTICES

- **Respond quickly** - Return 200 within 5 seconds
- **Process async** - Queue events for processing
- **Handle duplicates** - Events may be retried
- **Verify signatures** - Always validate authenticity
- **Log everything** - Keep webhook logs for debugging

ERROR HANDLING

Dealing with failures gracefully

HTTP STATUS CODES

Code	Meaning	Action
200	Success	Process response
202	Accepted	Event queued
400	Bad Request	Fix request data
401	Unauthorized	Check credentials
403	Forbidden	Check permissions
429	Rate Limited	Backoff and retry
500	Server Error	Retry with backoff
503	Unavailable	Retry later

ERROR RESPONSE FORMAT

```
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid phone number format",
    "details": {
      "field": "a_number",
      "value": "invalid",
      "expected": "E.164 format (+234XXXXXXXXX)"
    }
  },
  "meta": {
    "request_id": "req_abc124",
    "timestamp": "2026-01-15T10:30:00Z"
  }
}
```

Always include `request_id` in support tickets!

ERROR CODES

Code	Description
`VALIDATION_ERROR`	Invalid input data
`AUTH_FAILED`	Authentication failed
`PERMISSION_DENIED`	Insufficient permissions
`RESOURCE_NOT_FOUND`	Alert/resource doesn't exist
`RATE_LIMITED`	Too many requests
`INTERNAL_ERROR`	Server-side error

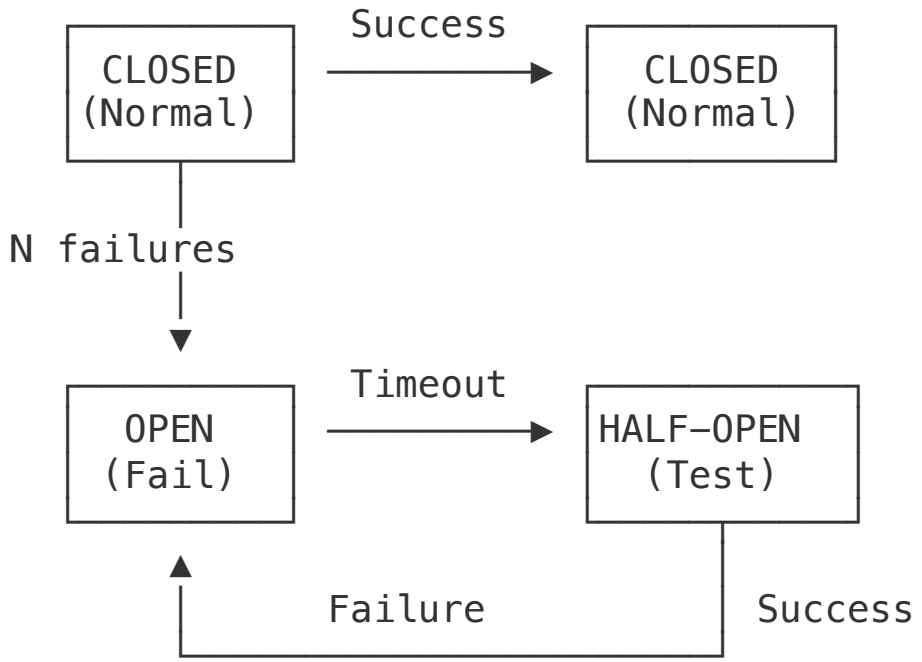
RETRY STRATEGY

```
async function submitWithRetry(event, maxRetries = 3) {
  for (let attempt = 1; attempt <= maxRetries; attempt++) {
    try {
      const response = await fetch(API_URL + '/events', {
        method: 'POST',
        headers: {
          'X-API-Key': API_KEY,
          'Content-Type': 'application/json'
        },
        body: JSON.stringify(event)
      });

      if (response.ok) {
        return await response.json();
      }
```

CIRCUIT BREAKER PATTERN

> Implement circuit breaker to avoid overwhelming a failing system



BEST PRACTICES

Building robust integrations

INTEGRATION CHECKLIST

- [ ] Use HTTPS for all API calls
- [ ] Store credentials securely (env vars/secrets manager
- [ ] Implement proper error handling
- [ ] Add retry logic with exponential backoff
- [ ] Validate webhook signatures
- [ ] Log all API interactions
- [ ] Monitor for rate limiting
- [ ] Test in sandbox before production

PERFORMANCE TIPS

- **Connection pooling** - Reuse HTTP connections
- **Batch requests** - Group events when possible
- **Async processing** - Don't block on API calls
- **Local caching** - Cache config/whitelist data
- **Compression** - Use gzip for large payloads

SECURITY CHECKLIST

- [ ] Never log sensitive data (API keys, tokens)
- [ ] Rotate API keys regularly
- [ ] Use minimal required scopes
- [ ] Validate all webhook payloads
- [ ] Use separate keys for prod/staging
- [ ] Implement request signing for high-security

TESTING STRATEGY

Environment	Purpose	Data
Sandbox	Development & testing	Synthetic
Staging	Integration testing	Anonymized
Production	Live traffic	Real

> Always test thoroughly in sandbox before deploying to production!

MONITORING YOUR INTEGRATION

KEY METRICS:

- API response times
- Error rates by type
- Rate limit headroom
- Webhook delivery success
- Event processing latency

Set up alerts for anomalies!

KEY TAKEAWAYS

- **Authentication** - API Keys or OAuth 2.0
- **Events** - Single or batch submission
- **Alerts** - Query, update, disconnect
- **Webhooks** - Real-time notifications
- **Errors** - Retry with backoff
- **Security** - Validate, encrypt, audit

RESOURCES

DOCUMENTATION:

- `API_DEVELOPER_MANUAL.md`
- `DEVELOPER_MANUAL.md`
- OpenAPI spec: `/api/v1/openapi.json`

SUPPORT:

- Slack: #api-support
- Email: api@yourcompany.c
- GitHub: Issues

QUESTIONS?

Developer Training Complete

Next Steps:

1. Get sandbox credentials
2. Run sample integrations
3. Build your first webhook