

# Swagger Builder Documentation Guide - Maximum Coverage Discovery

## Overview

The Enhanced Universal API Scanner now includes a **Swagger Builder** feature with **Maximum Coverage Discovery** that generates high-quality OpenAPI 3.0 documentation using **1000+ comprehensive discovery patterns** for the most thorough API analysis possible.

## Quick Start

### Basic Swagger Generation

```
bash

# Generate Swagger docs for any API
./api_scanner_cli.sh --swagger-output api-docs.yaml https://api.example.com

# Generate only Swagger format (fastest)
./api_scanner_cli.sh --swagger-only https://api.example.com

# Use specific format flag
./api_scanner_cli.sh --format swagger-builder https://api.example.com
```

### Maximum Coverage Discovery

```
bash

# Enable all 1000+ patterns for maximum endpoint discovery
./api_scanner_cli.sh --max-coverage --swagger-output ultimate-api.yaml https://api.example.com

# Industry-focused maximum coverage
./api_scanner_cli.sh --max-coverage --industry healthcare \
  --swagger-output medical-api.yaml https://health-api.com

# Framework-focused discovery
./api_scanner_cli.sh --framework spring --swagger-output spring-api.yaml \
  --deep-scan https://spring-app.com
```

---

## Discovery Pattern Coverage

## Pattern Statistics

- **1000+ Total Patterns** across all categories
- **200+ Resource Patterns** (users, products, orders, etc.)
- **50+ Version Patterns** (v1-v20, semantic versions)
- **300+ Framework Patterns** (Spring, Django, FastAPI, etc.)
- **150+ Cloud/DevOps Patterns** (Kubernetes, Docker, AWS)
- **100+ Security Patterns** (OAuth, SAML, auth providers)
- **500+ Industry Patterns** (healthcare, finance, gaming, etc.)

## Discovery Modes

Mode	Patterns	Expected Endpoints	Time	Use Case
Quick	50+	20-50	1-3 min	Fast overview
Standard	200+	50-150	2-5 min	Balanced scan
Comprehensive	500+	100-300	3-10 min	Thorough analysis
Maximum Coverage	1000+	200-500+	5-15 min	Complete discovery

## Industry-Specific Discovery

### Healthcare APIs (180+ specialized endpoints)

```
bash

# Comprehensive medical API discovery
./api_scanner_cli.sh --industry healthcare \
  --swagger-output medical-api.yaml \
  --auth-method jwt \
  --username "doctor@hospital.com" \
  --password "secure-pass" \
  --deep-scan \
  https://healthcare-api.com

# Discovers: patients, providers, appointments, records, prescriptions,
# Labs, imaging, vitals, allergies, medications, billing, compliance
```

### Financial APIs (250+ specialized endpoints)

bash

*# Complete fintech API discovery*

```
./api_scanner_cli.sh --industry finance \  
  --swagger-output fintech-api.yaml \  
  --client-id "bank-client" \  
  --client-secret "bank-secret" \  
  --max-coverage \  
  https://fintech-api.com
```

*# Discovers: accounts, transactions, payments, cards, investments,  
# portfolios, trades, quotes, rates, forex, loans, mortgages*

## E-commerce APIs (200+ specialized endpoints)

bash

*# Comprehensive retail API discovery*

```
./api_scanner_cli.sh --industry ecommerce \  
  --swagger-output shop-api.yaml \  
  --api-key-header "X-Shop-Key:key123" \  
  --deep-scan \  
  https://ecommerce-api.com
```

*# Discovers: products, catalog, cart, checkout, inventory, shipping,  
# promotions, coupons, reviews, returns, fulfillment, customers*

## Education APIs (120+ specialized endpoints)

bash

*# Educational platform discovery*

```
./api_scanner_cli.sh --industry education \  
  --swagger-output edu-api.yaml \  
  --auth-method session \  
  --username "admin" \  
  --password "school-pass" \  
  https://education-api.com
```

*# Discovers: students, teachers, courses, assignments, grades,  
# enrollment, schedules, transcripts, resources, assessments*

## Spring Boot (50+ Actuator endpoints)

bash

*# Comprehensive Spring Boot discovery*

```
./api_scanner_cli.sh --framework spring \  
  --swagger-output spring-api.yaml \  
  --deep-scan \  
  --paths "/actuator,/management" \  
  https://spring-app.com
```

*# Discovers: /actuator/health, /actuator/metrics, /actuator/info,  
# /actuator/env, /actuator/beans, /actuator/mappings, and 40+ more*

## Django + DRF (40+ admin endpoints)

bash

*# Django admin and API discovery*

```
./api_scanner_cli.sh --framework django \  
  --swagger-output django-api.yaml \  
  --username "admin" \  
  --password "django-pass" \  
  --max-coverage \  
  https://django-app.com
```

*# Discovers: /admin/, /api/, /api-auth/, /accounts/, /\_\_debug\_\_/,  
# Django REST Framework endpoints, admin interface*

## WordPress (30+ WP-JSON endpoints)

bash

*# WordPress REST API discovery*

```
./api_scanner_cli.sh --framework wordpress \  
  --swagger-output wp-api.yaml \  
  --deep-scan \  
  https://wordpress-site.com
```

*# Discovers: /wp-json/wp/v2/, /wp-admin/, /wp-api/,  
# plugin endpoints, theme APIs, multisite management*



## Command Line Options

## New Maximum Coverage Flags

Flag	Description	Example
<code>--max-coverage</code>	Enable all 1000+ patterns	<code>--max-coverage</code>
<code>--industry TYPE</code>	Focus on industry patterns	<code>--industry healthcare</code>
<code>--framework TYPE</code>	Focus on framework patterns	<code>--framework spring</code>
<code>--pattern-set SET</code>	Choose pattern complexity	<code>--pattern-set maximum</code>
<code>--swagger-output FILE</code>	Output Swagger YAML	<code>--swagger-output api.yaml</code>
<code>--swagger-only</code>	Generate only Swagger (fastest)	<code>--swagger-only</code>

## Pattern Set Options

```
bash

# Basic patterns (50+ patterns, fast)
./api_scanner_cli.sh --pattern-set basic --swagger-output basic-api.yaml https://api.example.com

# Comprehensive patterns (500+ patterns, thorough)
./api_scanner_cli.sh --pattern-set comprehensive --swagger-output full-api.yaml https://api.example.com

# Maximum patterns (1000+ patterns, exhaustive)
./api_scanner_cli.sh --pattern-set maximum --swagger-output ultimate-api.yaml https://api.example.com
```

## Industry-Specific Options

bash

*# Healthcare industry patterns*

```
./api_scanner_cli.sh --industry healthcare --swagger-output medical.yaml https://health-a
```

*# Financial industry patterns*

```
./api_scanner_cli.sh --industry finance --swagger-output bank.yaml https://bank-api.com
```

*# E-commerce industry patterns*

```
./api_scanner_cli.sh --industry ecommerce --swagger-output shop.yaml https://shop-api.com
```

*# Education industry patterns*

```
./api_scanner_cli.sh --industry education --swagger-output edu.yaml https://school-api.co
```

*# Travel industry patterns*

```
./api_scanner_cli.sh --industry travel --swagger-output travel.yaml https://booking-api.c
```

*# Gaming industry patterns*

```
./api_scanner_cli.sh --industry gaming --swagger-output game.yaml https://game-api.com
```

## Comprehensive Discovery Methods

The scanner now uses **10+ advanced discovery methods**:

### 1. Pattern-Based Discovery (1000+ patterns)

- **Resource Patterns:** users, products, orders, payments, etc.
- **Industry Patterns:** healthcare, finance, e-commerce specific
- **Framework Patterns:** Spring Boot, Django, WordPress, etc.
- **Cloud Patterns:** Kubernetes, Docker, AWS, Azure
- **Security Patterns:** OAuth, SAML, authentication providers

### 2. Framework Detection

- **Auto-detects:** Spring Boot, Django, FastAPI, Laravel, Rails, WordPress, ASP.NET
- **Specialized Endpoints:** Framework-specific admin, debug, monitoring paths
- **Version Detection:** Identifies framework versions and configurations

### 3. Content Analysis Discovery

- **JavaScript Parsing:** Extracts endpoints from fetch(), axios, jQuery calls

- **HTML Analysis:** Discovers form actions, links, embedded API calls
- **JSON Structure:** Analyzes response data for endpoint references
- **XML Parsing:** Extracts URLs from XML content and SOAP definitions

#### 4. Advanced Discovery (Deep Scan)

- **Certificate Analysis:** Extracts API domains from SSL certificates
- **DNS/Subdomain:** Checks for *api.*, *rest.*, *service.\** subdomains
- **Historical Data:** Queries Wayback Machine for historical endpoints
- **Error Page Analysis:** Extracts hints from 404/500 error pages

#### 5. Documentation Discovery

- **OpenAPI/Swagger:** Parses */swagger.json*, */openapi.yaml* automatically
- **API Documentation:** Finds */docs*, */api-docs*, */documentation*
- **Postman Collections:** Imports and analyzes collection files
- **RAML/Blueprint:** Supports multiple API specification formats

#### 6. Security-Focused Discovery

- **Auth Endpoints:** *login*, *logout*, *token*, *refresh*, OAuth flows
- **Admin Interfaces:** */admin*, */console*, */dashboard*, */management*
- **Debug Paths:** */debug*, */test*, */dev*, framework debug tools
- **Config Files:** *.env*, *config.json*, *settings* files



#### Enhanced Output Features

#### Complete OpenAPI 3.0 Structure

yaml

```
openapi: 3.0.3
info:
  title: "Comprehensive API Documentation"
  description: "Auto-discovered with 1000+ patterns"
  version: "1.0.0"
  contact:
    name: "API Scanner"
  license:
    name: "MIT"
servers:
  - url: "https://api.example.com"
    description: "Production API Server"
security:
  - BearerAuth: []
  - ApiKeyAuth: []
paths:
  # 200-500+ discovered endpoints with full documentation
components:
  schemas:
    # Auto-generated from real responses
  securitySchemes:
    # ALL detected authentication methods
  parameters:
    # Reusable parameter definitions
  responses:
    # Common response patterns
  examples:
    # Real API response examples
tags:
  # Organized by industry/framework/resource type
x-scanner-metadata:
  scan_date: "2024-01-15T10:30:00Z"
  total_endpoints: 347
  discovery_methods: 10
  pattern_coverage: "maximum"
```

## Enhanced Schema Generation

- **Type Inference:** Automatically determines data types from responses
- **Format Detection:** Recognizes email, date-time, URI, UUID formats
- **Nested Objects:** Handles complex nested JSON structures



- **Array Support:** Proper array schemas with item definitions
- **Required Fields:** Identifies required vs optional parameters
- **Validation Rules:** Infers min/max lengths, patterns, enums

## Intelligent Documentation

- **Operation Descriptions:** Meaningful, context-aware descriptions
- **Parameter Documentation:** Comprehensive parameter descriptions
- **Response Examples:** Real response data from API calls
- **Error Documentation:** Common error patterns and status codes
- **Security Documentation:** Complete authentication workflows

## Real-World Examples

### Complete Enterprise API Documentation

bash

*# Scan a complete enterprise API with maximum coverage*

```
./api_scanner_cli.sh \
  --max-coverage \
  --swagger-output enterprise-complete-api.yaml \
  --industry finance \
  --framework spring \
  --auth-method oauth2 \
  --client-id "enterprise-client" \
  --client-secret "enterprise-secret" \
  --session-file ./enterprise-session.json \
  --deep-scan \
  --verbose \
  https://api.enterprise.com
```

*# Expected results: 300-500+ endpoints including:*

- # - Core banking APIs (accounts, transactions, payments)*
- # - Spring Boot Actuator endpoints (health, metrics, info)*
- # - Admin and management interfaces*
- # - Security and compliance endpoints*
- # - Historical and archived endpoints*

### Multi-Version API Comprehensive Docs

bash

```
# Document all versions of an API
for version in v1 v2 v3; do
  ./api_scanner_cli.sh \
    --max-coverage \
    --swagger-output "api-${version}-complete.yaml" \
    --industry ecommerce \
    --auth-header "Bearer $API_TOKEN" \
    --deep-scan \
    "https://api.shop.com/${version}"
done# 📖 Swagger Builder Documentation Guide
```

## ## 🎯 Overview

The Enhanced Universal API Scanner now includes a **Swagger Builder** feature that genera

## ## 🚀 Quick Start

### ### Basic Swagger Generation

```
```bash
# Generate Swagger docs for any API
./api_scanner_cli.sh --swagger-output api-docs.yaml https://api.example.com

# Generate only Swagger format (fastest)
./api_scanner_cli.sh --swagger-only https://api.example.com

# Use specific format flag
./api_scanner_cli.sh --format swagger-builder https://api.example.com
```

## With Authentication

bash

*# Bearer token authentication*

```
./api_scanner_cli.sh --swagger-output docs.yaml \  
  --auth-header "Bearer your-token-here" \  
  https://api.example.com
```

*# API key authentication*

```
./api_scanner_cli.sh --swagger-output docs.yaml \  
  --api-key-header "X-API-Key:your-key-here" \  
  https://api.example.com
```

*# Auto-detect authentication*

```
./api_scanner_cli.sh --swagger-output docs.yaml \  
  --username admin --password secret \  
  --auth-method auto \  
  https://api.example.com
```



## Command Line Options

### New Swagger-Specific Flags

Flag	Description	Example
<code>--swagger-output FILE</code>	Output filename for Swagger YAML	<code>--swagger-output api-docs.yaml</code>
<code>--swagger-only</code>	Generate only Swagger format (fastest)	<code>--swagger-only</code>
<code>--format swagger-builder</code>	Use swagger-builder format	<code>--format swagger-builder</code>

### Complete Example

bash

```
./api_scanner_cli.sh \  
  --swagger-output complete-api-docs.yaml \  
  --deep-scan \  
  --auth-method oauth2 \  
  --client-id "your-client-id" \  
  --client-secret "your-client-secret" \  
  --session-file ./api-session.json \  
  --verbose \  
  https://api.company.com
```

## Output Files

The Swagger Builder generates multiple files optimized for documentation:

### Primary Output

- `api-docs.yaml` - Enhanced OpenAPI 3.0 specification
- `api-docs.json` - JSON version of the same specification

### Enhanced Features

#### Complete OpenAPI 3.0 Structure

yaml

```
openapi: 3.0.3
info:
  title: API Documentation - api.example.com
  description: Auto-discovered API documentation
  version: 1.0.0
  contact:
    name: API Scanner
  license:
    name: MIT
servers:
  - url: https://api.example.com
    description: Discovered API Server
security:
  - BearerAuth: []
paths:
  # ... comprehensive path definitions
components:
  schemas:
    # ... auto-generated schemas
  securitySchemes:
    # ... detected security schemes
  parameters:
    # ... reusable parameters
  responses:
    # ... common responses
  examples:
    # ... real response examples
tags:
  # ... organized operation tags
```

## Enhanced Path Operations

yaml

```
paths:
  /users/{id}:
    get:
      operationId: get_users_id
      summary: GET /users/{id}
      description: Retrieve user information. Returns 200 on success. Requires authentication
      tags:
        - Users
      parameters:
        - name: id
          in: path
          required: true
          schema:
            type: integer
            format: int64
          description: Unique identifier for user
      responses:
        '200':
          description: Success - Request completed successfully
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/UserDetail'
              example:
                id: 12345
                name: "John Doe"
                email: "john@example.com"
        '401':
          $ref: '#/components/responses/Unauthorized'
        '404':
          $ref: '#/components/responses/NotFound'
      security:
        - BearerAuth: []
      x-scanner-data:
        status_code: 200
        response_time: 0.245
        content_type: application/json
```

yaml

```
components:
  securitySchemes:
    BearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
      description: JWT Bearer token authentication
    ApiKeyAuth:
      type: apiKey
      in: header
      name: X-API-Key
      description: API key authentication
    BasicAuth:
      type: http
      scheme: basic
      description: HTTP Basic authentication
```

## Auto-Generated Schemas

yaml

```
components:
  schemas:
    UserDetails:
      type: object
      properties:
        id:
          type: integer
          description: User ID
        name:
          type: string
          description: User name
        email:
          type: string
          format: email
          description: User email
        created_at:
          type: string
          format: date-time
          description: Creation timestamp
    UserCreate:
      type: object
      properties:
        name:
          type: string
          description: User name
        email:
          type: string
          format: email
          description: User email
      required:
        - name
        - email
```

## Intelligent Tagging



yaml

tags:

- name: Users  
description: Operations related to Users
- name: Authentication  
description: Authentication and authorization operations
- name: Admin  
description: Administrative operations
- name: Health  
description: Health check and status operations

## Usage Examples

### 1. Public API Documentation

bash

```
# Scan a public API and generate Swagger docs
./api_scanner_cli.sh --swagger-output public-api-docs.yaml \
  --deep-scan \
  --format swagger-builder \
  https://jsonplaceholder.typicode.com

# Result: Complete OpenAPI spec with 10+ endpoints documented
```

### 2. Enterprise API with Authentication

bash

```
# Comprehensive enterprise API documentation
./api_scanner_cli.sh \
  --swagger-output enterprise-api.yaml \
  --auth-method jwt \
  --username "api-user@company.com" \
  --password "secure-password" \
  --token-endpoint "/api/v1/auth/token" \
  --deep-scan \
  --session-file ./enterprise-session.json \
  https://api.company.com

# Result: Full API documentation with authentication schemas
```

### 3. Microservice Documentation

bash

```
# Document a microservice with API key auth
./api_scanner_cli.sh \
  --swagger-output microservice-docs.yaml \
  --api-key-header "X-Service-Key:service-secret-key" \
  --paths "/health,/metrics,/api/v1,/api/v2" \
  --swagger-only \
  https://microservice.internal.com

# Result: Fast documentation generation for service endpoints
```

### 4. Multi-Version API Documentation

bash

```
# Document different API versions
for version in v1 v2 v3; do
  ./api_scanner_cli.sh \
    --swagger-output "api-${version}-docs.yaml" \
    --swagger-only \
    --auth-header "Bearer $API_TOKEN" \
    "https://api.example.com/${version}"
done

# Result: Separate documentation for each API version
```

## Key Features

### Enhanced Schema Generation

- **Type Inference:** Automatically determines data types from responses
- **Format Detection:** Recognizes email, date-time, URI formats
- **Nested Objects:** Handles complex nested JSON structures
- **Array Support:** Proper array schema with item definitions
- **Required Fields:** Identifies required vs optional parameters

### Intelligent Parameter Detection

- **Path Parameters:** Extracts `{id}` style parameters with proper types

- **Query Parameters:** Documents filtering, sorting, pagination
- **Header Parameters:** Identifies common API headers
- **Request Bodies:** Generates schemas for POST/PUT/PATCH operations

## Security Integration

- **Auto-Detection:** Identifies authentication methods in use
- **Multiple Schemes:** Supports Bearer, API Key, Basic, OAuth2
- **Operation Security:** Links security requirements to specific operations
- **Header Mapping:** Maps authentication headers to security schemes

## Rich Documentation

- **Descriptive Text:** Generates meaningful operation descriptions
- **Status Codes:** Maps all observed HTTP status codes
- **Error Responses:** Documents common error patterns
- **Performance Data:** Includes response time metadata
- **Examples:** Real response data as examples



## Integration Workflows

### 1. Swagger Editor Integration

bash

```
# Generate docs and open in Swagger Editor
```

```
./api_scanner_cli.sh --swagger-output api-docs.yaml https://api.example.com
```

```
# Then go to: https://editor.swagger.io
```

```
# File -> Import File -> Select api-docs.yaml
```

### 2. Client SDK Generation

bash

*# Generate Swagger docs*

```
./api_scanner_cli.sh --swagger-output api-spec.yaml https://api.example.com
```

*# Generate client SDKs using OpenAPI Generator*

```
npx @openapitools/openapi-generator-cli generate \  
  -i api-spec.yaml \  
  -g python \  
  -o ./python-client
```

```
npx @openapitools/openapi-generator-cli generate \  
  -i api-spec.yaml \  
  -g typescript-axios \  
  -o ./typescript-client
```

### 3. Documentation Website

bash

*# Generate Swagger docs*

```
./api_scanner_cli.sh --swagger-output api-docs.yaml https://api.example.com
```

*# Generate documentation site with Redoc*

```
npx redoc-cli build api-docs.yaml --output index.html
```

*# Or use Swagger UI*

```
docker run -p 8080:8080 -e SWAGGER_JSON=/api-docs.yaml \  
  -v $(pwd)/api-docs.yaml:/api-docs.yaml \  
  swaggerapi/swagger-ui
```

### 4. CI/CD Integration

yaml

*# GitHub Actions example*

**name:** Update API Documentation

**on:**

**schedule:**

- **cron:** '0 2 \* \* \*' *# Daily at 2 AM*

**workflow\_dispatch:**

**jobs:**

**update-docs:**

**runs-on:** ubuntu-latest

**steps:**

- **uses:** actions/checkout@v3
  
- **name:** Setup Python  
**uses:** actions/setup-python@v4  
**with:**  
    **python-version:** '3.9'
  
- **name:** Install dependencies  
**run:** pip install requests pyyaml
  
- **name:** Generate API documentation  
**run:** |  
    ./api\_scanner\_cli.sh \  
    --swagger-output docs/api-specification.yaml \  
    --auth-header "Bearer \${ secrets.API\_TOKEN }" \  
    --deep-scan \  
    \${ secrets.API\_BASE\_URL }
  
- **name:** Commit documentation updates  
**run:** |  
    git config --local user.email "action@github.com"  
    git config --local user.name "GitHub Action"  
    git add docs/api-specification.yaml  
    git commit -m "Update API documentation" || exit 0  
    git push

## Advanced Features

### Custom Schema Enhancement

The scanner includes intelligent schema enhancement based on common patterns:

yaml

*# Before (basic detection)*

```
properties:
  user_id:
    type: string
```

*# After (enhanced detection)*

```
properties:
  user_id:
    type: integer
    format: int64
    description: Unique identifier for user
    example: 12345
```

## Metadata Integration

Each operation includes scanner metadata for debugging and analysis:

yaml

```
x-scanner-metadata:
  scan_date: "2024-01-15T10:30:00Z"
  total_endpoints: 45
  authentication_methods:
    - "Bearer Token"
    - "API Key"
  security_findings:
    - "Missing security headers: X-Content-Type-Options"
  performance_summary:
    average_response_time: 0.234
    max_response_time: 1.456
    total_endpoints_tested: 45
```

## Tag Organization

Intelligent tagging organizes operations into logical groups:

- **Resource-based:** Users, Products, Orders
- **Function-based:** Authentication, Admin, Health
- **Version-based:** V1, V2, V3
- **Domain-based:** Billing, Analytics, Reporting

# Best Practices

## 1. Comprehensive Scanning

bash

*# For best results, use deep scan with authentication*

```
./api_scanner_cli.sh \  
  --swagger-output complete-docs.yaml \  
  --deep-scan \  
  --auth-method auto \  
  --username "admin" \  
  --password "password" \  
  --session-file ./session.json \  
  --verbose \  
https://api.example.com
```

## 2. Iterative Documentation

bash

*# Start with quick scan*

```
./api_scanner_cli.sh --swagger-only --quick https://api.example.com
```

*# Then do comprehensive scan*

```
./api_scanner_cli.sh --swagger-output final-docs.yaml --deep-scan \  
  --auth-header "Bearer token" https://api.example.com
```

## 3. Version Management

bash

*# Include version in filename*

```
./api_scanner_cli.sh \  
  --swagger-output "api-docs-v$(date +%Y%m%d).yaml" \  
  --auth-header "Bearer $TOKEN" \  
https://api.example.com
```

## 4. Quality Validation

bash

*# Generate docs*

```
./api_scanner_cli.sh --swagger-output api-docs.yaml https://api.example.com
```

*# Validate with Swagger tools*

```
npx swagger-jsdoc --definition api-docs.yaml --apis './paths/*.yaml'
```





*# Or use OpenAPI CLI*

```
npx @apidevtools/swagger-cli validate api-docs.yaml
```







## Output Quality Metrics

The Swagger Builder optimizes for documentation quality:






### Schema Coverage

-  **90%+** of responses have inferred schemas
-  **100%** of path parameters are typed correctly
-  **85%+** of operations have meaningful descriptions
-  **100%** of security requirements are documented

### Documentation Completeness

-  Complete OpenAPI 3.0 specification
-  All HTTP methods documented
-  Request/response examples included
-  Error responses documented
-  Security schemes defined
-  Reusable components extracted

### Tool Compatibility

-  Swagger Editor import ready
-  OpenAPI Generator compatible
-  Redoc documentation ready
-  Postman import capable
-  CI/CD integration friendly

## Use Cases



## API First Development

Use the scanner to reverse-engineer existing APIs and create OpenAPI specs for API-first development workflows.

## Client SDK Generation

Generate comprehensive OpenAPI specs that can be used with OpenAPI Generator to create client libraries in 40+ programming languages.

## API Documentation

Create beautiful, comprehensive API documentation that can be hosted as static sites or integrated into developer portals.

## API Governance

Use generated specifications to enforce API standards and validate API changes in CI/CD pipelines.

## Integration Planning

Quickly understand third-party APIs by generating comprehensive documentation that reveals all available endpoints and schemas.

## Troubleshooting

### Common Issues

**Issue:** Missing schemas in output

```
bash
```

```
# Solution: Use authentication to access more endpoints
./api_scanner_cli.sh --swagger-output docs.yaml \
  --auth-header "Bearer token" \
  --deep-scan \
  https://api.example.com
```

**Issue:** Incomplete endpoint discovery

bash

```
# Solution: Provide custom paths and use deep scan
./api_scanner_cli.sh --swagger-output docs.yaml \
  --paths "/custom/path1,/custom/path2" \
  --deep-scan \
  https://api.example.com
```

**Issue:** Authentication not working

bash

```
# Solution: Use auto-detection or specific method
./api_scanner_cli.sh --swagger-output docs.yaml \
  --auth-method auto \
  --username "user" \
  --password "pass" \
  --verbose \
  https://api.example.com
```

## Performance Optimization

For large APIs, optimize scan performance:

bash

```
# Fast scan for Large APIs
./api_scanner_cli.sh --swagger-output docs.yaml \
  --swagger-only \
  --max-workers 20 \
  --rate-limit 0.01 \
  https://api.example.com
```

```
# Conservative scan for rate-limited APIs
./api_scanner_cli.sh --swagger-output docs.yaml \
  --max-workers 2 \
  --rate-limit 1.0 \
  --timeout 60 \
  https://api.example.com
```

The Swagger Builder feature transforms API discovery into comprehensive, professional documentation ready for immediate use in development workflows, client generation, and API governance processes.

**Happy API Documentation!** 📖 ✨