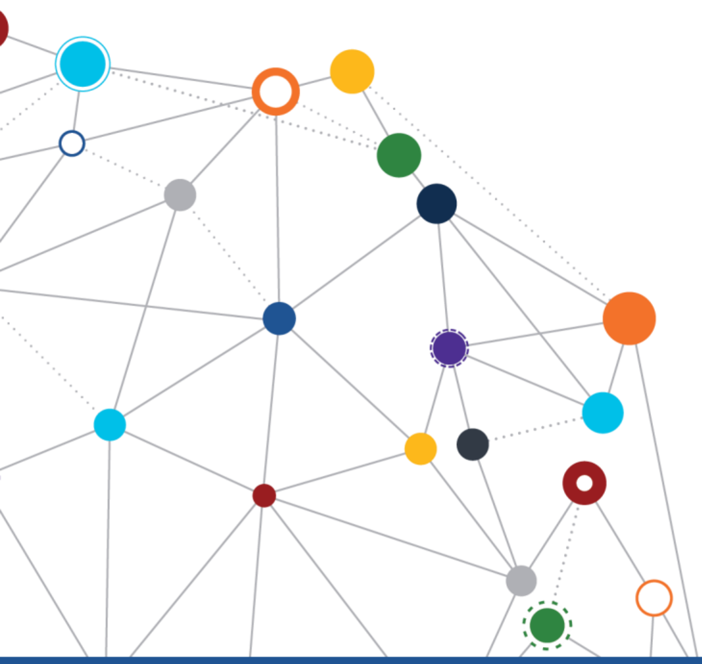
OFFICE OF
INFORMATION
AND TECHNOLOGY

Application Programming Interface (API) Enterprise Design Pattern

Documenting API Security

February 2019 | Enterprise Program Management Office



VA | U.S. Department of Veterans Affairs
Office of Information and Technology

Table of Contents

Table 1: Change Matrix

Version	Date	Description of Updates
1.0	10/03/18	API Segment: Documenting API Security document approved
2.0	2/22/19	Added Open API field details and additional references

1 Context

The Department of Veterans Affairs (VA) implements numerous information technology (IT) projects with application programming interfaces (APIs) that expose underlying services. When documenting an API, VA project teams should consider *security actions that ensure the secure use of APIs in the enterprise*. This document provides recommendations to project teams that are based on API features and Open API security, providing details that supplement the *API Documentation Standard* Enterprise Design Pattern (EDP).[1] The key recommendations are not a complete set of guidelines for all API security.[2]

2 Problem

VA APIs span numerous use cases and expose VA systems to various types of security vulnerabilities (e.g., data compromise, improper authentication, and "man in the middle" attacks). To ensure that systems are used securely by end users and operators, project teams should *plan for API security and documentation of security features*.

3 Approach

In conjunction with the *API Documentation Standard* EDP, this document provides project teams with actionable steps and planning recommendations to resolve security concerns when documenting APIs. Additional information related to these topics can be found in the *Identity and Access Management* (IAM) EDPs. This four-part approach entails (1) determining security needs and features, (2) determining the OpenAPI security mechanisms, (3) documenting the OpenAPI security content, and (4) addressing additional security settings.

3.1 Determine Security Needs and Features of the API

Initially, project teams need to determine their API and system security needs because these will influence the security mechanisms selected and detailed in section 3.2. Project teams need to identify and examine appropriate enterprise information resources during the API development phase.

[1] Refer to the *API Documentation Standard* Enterprise Design Pattern (EDP), June 2018, https://www.oit.va.gov/library/files/edp/apis/APIEDP_DocumentationStandard_v1.pdf.
[2] For additional information, refer to the *VA Application Programming Interface Security Pattern*, Cybersecurity Architecture Office, June 2018 at https://vaww.vashare.oit.va.gov/sites/ois/KnowledgeService/SecurityDocuments/Cybersecurity%20Architecture%20Docs/VA_API_Security_Pattern_v1.2.pdf; also refer to API EDPs at https://www.oit.va.gov/library/recurring/edp/.

Security requirements differ according to the use of the API. Key considerations to determine the security requirements include:

- The type of information and service offered by the API and the impact of loss or vulnerabilities to unauthorized entities or adversaries[3]
- Threats from API users or adversaries
- Compliance requirements, which may include the VA IAM directive, IAM handbook, Veteran-Focused Integration Process (VIP),[4] user stories,[5] EDPs, and security patterns
- The authentication requirements for the API end user
- The necessary authorization before a client application (software) can access an API

The following subsections provide details to help with the first category of considerations and for the documentation of each API. The steps for planning the processing of sensitive information, open data, and backend services help determine security mechanisms needed. Further security details are subject to the VA Security and Privacy EDPs[6] and the VA API Security Pattern.[7]

3.1.1 When Processing Sensitive Information

There are different types of data exposed in a VA ecosystem that may be categorized as sensitive data, such as personally identifiable information (PII), protected health information (PHI), payment card information (PCI), and other sensitive financial data.

The first step in determining proper security controls (technical, managerial, or operational) for sensitive data is for project teams to identify the requirements through an analysis of the inbound and outbound data to the API (Refer to the Veteran-Focused Integration Process (VIP) Security Guide, November 2017).[8] The VIP Security Guide[9] provides procedures for identifying

[3] Project teams should request and implement data use requirements provided by the corresponding data steward(s).
[4] Refer to the Veteran-Focused Integration Process (VIP) 3.2 Guide, December 2018, at https://vaww.vaco.portal.va.gov/sites/OIT/epmo/vip/Policy%20%20Guidance/VIP%20Guide%203.2.pdf.
[5] For VA Design, Engineering, and Architecture (DEA) User Stories, refer to version 2.3, October 11, 2018, at https://vaww.portal2.va.gov/sites/asd/TechStrat/IPTS/External%20Documents/DEA%20User%20Stories%20v2.3%20(ACTIVE).doc.
[6] Refer to the Security and Privacy EDPs at https://www.oit.va.gov/library/recurring/edp/index.cfm.
[7] Refer to the VA API Security Pattern, Cybersecurity Architecture Office, June 2018 at https://vaww.vashare.oit.va.gov/sites/ois/KnowledgeService/SecurityDocuments/Cybersecurity%20Architecture%20Docs/VA_API_Security_Pattern_v1.2.pdf.
[8] Refer to the VA Office of Information and Technology (OIT) Office of Cyber Security (OCS) Enterprise Security Architect (ESA) Security Guide to Veteran-Focused Integration Process (VIP), Version 1.0, November 2017, at https://vaww.ea.oit.va.gov/wp-content/uploads/2018/08/20171121_VIP_Security_Guide_1.0.pdf.
[9] Ibid.

the information types to understand the sensitivity of data and information, and to determine how to integrate security within an information system/application that is already in operation.

The following security requirements **must** be considered when documenting APIs that process sensitive information:

- Non-public Representational State Transfer (REST) services **must** perform access control at each API endpoint. Web services in monolithic applications implement this by means of user authentication, authorization logic (e.g., using a message authorization header), and session management.
- External service provider APIs should go through a VA-approved API gateway solution.
- All sensitive information **must** be encrypted in transit using Transport Layer Security (TLS) 1.2 or higher. Implementations may also opt to use Internet Protocol Security (IPsec).

3.1.2 When Processing Open Data

Open Data APIs (e.g., Veteran and non-Veteran interments at National cemeteries) are defined as *public-facing resource endpoints that can be consumed with no restrictions by the developer community at large, inside and outside of an organization*.[10] While the data may be unrestricted, the API implementation **must** have security and quality controls to protect the underlying system and ensure integrity, performance, and availability. VA's Open Data APIs are made available through the VA Open Data Portal,[11] with certain exceptions that are based on OMB Memorandum 13-13.[12]

Project teams should consider the following security requirements when documenting Open Data APIs:

- VA Open Data APIs **must** follow guidelines in accordance with OMB Memorandum 13-13.[13] Based on this memo, projects **must** limit sharing of open data sets with privacy concerns (e.g., PHI, PII), or legal restrictions.[14] Where possible, sensitive data **must** be

[10] In this document, Open Data APIs refer to APIs that provide access to open data sets, which can be distributed without restriction. Open API refers to the Open API specification for describing APIs; this specification is available at https://github.com/OAI/OpenAPI-Specification.
[11] Refer to the VA Open Data Portal at https://www.data.va.gov/. Some APIs restrict public access; developers are required to contact the API Owner for access to the API, through a registration key.
[12] Refer to Office of Management and Budget (OMB) Memorandum 13-13 at https://obamawhitehouse.archives.gov/sites/default/files/omb/memoranda/2013/m-13-13.pdf
[13] Ibid.
[14] For additional information, refer to OMB Memorandum 13-13, page 10 at https://obamawhitehouse.archives.gov/sites/default/files/omb/memoranda/2013/m-13-13.pdf.

protected through implementation of data masking and all API actions **must** be logged for auditing; it is recommended that tokenization[15] be used for payment card information. Also, where possible, openness should be favored, and implementation of authorization and authentication controls should be avoided.[16]

When implementing the underlying service:

- Traffic throttling[17] should be used so that clients who are downloading data in bulk through an open API (a common occurrence) do not have an adverse impact on other clients, or increase cloud consumption.

3.1.3 When Invoking Backend Services

Project teams should refer to the supported security schemes outlined in the OpenAPI specification for security of backend services.[18] Details on security schemes for open data can be found in Section 3.2 of this document.

Identification, authentication, and authorization, among other topics, should be addressed when documenting API security features, since they are concerns for data interactions and storage (e.g., guard APIs by accessing and masking sensitive encrypted data at runtime and protecting backend services against direct access). This effort also requires aligning the API and service with the VA API gateway.[19] Project teams should consult *IAM* EDPs for further guidance on access controls.[20]

Project teams should document security for **data interaction** by implementing the following:

- *Identification*: Assess the applications that will interface with the API to understand the identification use cases.[21]

[15] Tokenization is the process of replacing sensitive data with unique identification symbols that retain all the essential information about the data without compromising its security. For more information, refer to https://en.wikipedia.org/wiki/Tokenization_(data_security).
[16] When needed, project teams should recognize that open data allows for the laxest security controls. Projects may consider HTTP Basic Authentication. HTTP user agent provides a username and password to prove their authentication.
[17] For more information on throttling, refer to https://nordicapis.com/safely-throttle-high-traffic-apis/.
[18] Refer to the OpenAPI Specification Version 3.0.2 at https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.2.md.
[19] API gateway security will be covered in a future document.
[20] Refer to EDPs at https://www.oit.va.gov/library/recurring/edp/.
[21] Source: Greg Brail. APIs for Data Security and Privacy: Part One - Apigee's testimony before the ONC API Task Force. https://apigee.com/about/blog/digital-business/apis-data-security-and-privacy-part-one-0.

- *Authorization and Authentication:* Refer to the latest OAuth protocol as part of the planning for secure API Authorization.[22] Refer to the VA Technical Reference Model (TRM) guidelines for corresponding usage restrictions.[23] Consult the *IAM* EDP as part of the planning for secure API Authentication.[24]
- *Traffic Management:*[25] Determine the rate at which the API is used by each application and usage policies, such as quota and spike arrest to mitigate disruptions to the backend. [26]
- *Threat Detection:* Consult the Veteran-focused Integration Process (VIP) Security Guide for guidance on security modeling.[27] This includes guidance on threat modeling and applying the spoofing, tampering, and repudiation information disclosure, denial-of-service, and the escalation of privileges (STRIDE) technique.[28]

Project teams should document security for **data storage** by adhering to NIST Special Publication (SP) 800-175B for guidance on encryption for data at rest.[29]

3.2 Determine Open API Security Mechanisms

Based on considerations including those in section 3.1, project teams **must** select appropriate security mechanisms. These mechanisms should control user access to APIs and/or the applications calling the APIs. Table 2 provides information that can help in selecting the appropriate Open API security mechanisms.

[22] Refer to The OAuth web site at https://oauth.net/2/.
[23] Refer to the VA Technical Reference Model (TRM) to identify approved applications and standards on the internal VA network at http://trm.oit.va.gov/. External vendors may utilize a less comprehensive site at https://www.oit.va.gov/services/trm/.
[24] Refer to the IAM EDP at https://www.oit.va.gov/library/recurring/edp/.
[25] Source: Greg Brail, APIs for Data Security and Privacy: Part One; Apigee's testimony before the ONC API Task Force at https://apigee.com/about/blog/digital-business/apis-data-security-and-privacy-part-one-0.
[26] Source: Rakesh Talanki and Vidheer Gadikota. *Best Practices for Building Secure APIs* at https://medium.com/apis-and-digital-transformation/best-practices-for-building-secure-apis-2b4eb8071d41.
[27] Refer to the VIP Guide at https://vaww.vaco.portal.va.gov/sites/OIT/epmo/vip/Policy%20%20Guidance/VIP%20Guide%203.2.pdf.
[28] STRIDE was developed by Microsoft Corporation. Additional information is available at https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20).
[29] Source: National Institute of Standards and Technology (NIST) Special Publication 800-175B, *Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms,* at http://dx.doi.org/10.6028/NIST.SP.800-175B.

Table 2: Security Schemes from the Open API Specification

Type	Description	Comments
http (Hypertext Transfer Protocol)	Defines web authentication using a username and password. This includes HTTP basic and bearer and other HTTP authentications schemes (in field `scheme`)	• Requires Hypertext Transfer Protocol Secure (HTTPS); e.g., Transport Layer Security (TLS) to protect credentials[30] • Requires compliance with VA IAM policy, handbook, and services[31] • Requires the scheme to be defined: basic, bearer, or other • Basic authentication should be used when the Application Programming Interface (API) needs to identify the end user
apiKey[32]	Defines API keys in query, headers, or cookies API keys are used to identify the software that invokes the API and service (i.e., the keys identify the calling or consuming software); these are used only for non-sensitive and read-only data. API keys typically do not identify individual users	• **Must** follow API key best practices, including encryption while at rest and in transit; the key is not otherwise encrypted and does not have a signature itself • Requires TLS, which encrypts the path and parameters to protect sensitive information and the API key • Requires location of API key to be defined (using `in` field) • Requires name of the header, query parameter, or cookie to be defined (using `name` field)

[30] Without HTTPS, credentials may only go through a base64 encoding, which does not provide a level of confidentiality beyond obscuring the data. However, usage without HTTPS can be necessary or useful when serving a large community of end users and when distributing client certificates is not feasible (e.g., serving FOIA request information).

[31] Refer to the VA Directive 6510, *VA Identity and Access Management*, January 15, 2016 and VA Handbook 6510, "VA Identity and Access Management," January 15, 2016.

[32] An example of this security mechanism can be found with both the *Benefits Intake* and *Facilities* API.

Type	Description	Comments
		• Where the system and client can support it, use the HTTP header to pass the API key[33] • To conform to the RESTful stateless architecture style, avoid use of cookies for the API key • API keys should not be passed in the Uniform Resource Locator (URL)[34]
oauth2	Defines OAuth version 2.0 This is standard for a process for authorization, whereby third-party applications can access an application or service on behalf of another user API management platforms can implement these features for projects Used by mobile and native applications, especially third-party, untrusted applications, so that the user does not need to share credentials with the applications	• Requires compliance with VA IAM policy, handbook, and services[35] • OAuth provides an authorization process and depends on another service for authentication, such as Security Assertion Markup Language (SAML), or OpenID Connect (OIDC) • See *IAM OAuth* Enterprise Design Pattern (EDP) segments for more details on OAuth design and grant selection • Requires scopes to be defined • Requires "flow" or grant type to be defined • Requires authorization and refresh URLs to be defined • May require a token URL to be defined

[33] Refer to the Internet Engineering Task Force (IETF) specification at https://tools.ietf.org/html/rfc7235.
[34] The uniform resource locator (URL), including the key and parameters, may be saved on the client side, server side, or elsewhere.
[35] Refer to VA Directive 6510, *VA Identity and Access Management*, January 15, 2016. and VA Handbook 6510, *VA Identity and Access Management*, January 15, 2016.

Type	Description	Comments
openIdConnect[36]	Defines OpenID Connect version 1.0 Serves as an authentication layer on top of OAuth Useful when end user authentication is needed	• OIDC use is constrained for version 1.0 and unapproved for 2.0 in the VA Technical Reference Model (TRM) • Requires compliance with VA IAM policy, handbook, and services[37] • Implemented by VA API Developer Platform using Okta • Not currently supported by VA IAM services

VA project teams should document the API using a text editor in accordance with the format, structure, parameters, and guidelines of the latest version of the Open API specification[38] and the *API Documentation Standard* EDP.[39] Project teams **must** place these materials in the API Developer Portal and ESCP.[40]

The project team creating the API should also document steps that service consumers can follow to securely use the API.[41] For examples steps may include:

- Method for registration to obtain an API key.[42]
- Process to sign up for a production token.
- References for additional details on implementing the secure mechanisms.[43]

3.3 Document the Open API Security Content

When documenting using the Open API specification, project teams should define the security objects listed in the following subsections as tailored to the security features the API will

[36] An example includes the Veterans Health API (e.g., a system using SMART on FHIR).
[37] Refer to VA Directive 6510, *VA Identity and Access Management*, January 15, 2016, and VA Handbook 6510, *VA Identity and Access Management*, January 15, 2016.
[38] Source: OpenAPI Specification Version 3.0 at https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md.
[39] Refer to the API Documentation Standard, June 2018 at https://www.oit.va.gov/library/files/edp/apis/APIEDP_DocumentationStandard_v1.pdf.
[40] For research purposes, additional information on APIs is also available in the VEAR at https://vaausdarapp82.aac.dva.va.gov/ee/request/home.
[41] Additional information such as these instructions, can be added to the Open API External Documentation Object.
[42] To obtain an API key, register at https://developer.va.gov/apply.
[43] For example, details on using OAuth with the API Developer Portal are found at https://developer.va.gov/oauth.

implement. In documenting the API's security features, the documentation **must** include specifying the schema for Security Scheme Object, OAuth Flows Object, and Security Requirement Object, which define the security mechanisms that can be used. These are described in Table 3.

Tools are available to assist with creating these definitions and with validating that your Open API documentation meets the Open API specification.[44]

Table 3: Open API Fields

Open API Field	Description	Location (Field Names)
securitySchemes	Defines the security scheme or schemes used by the API (e.g., HTTP authentication, API key, OAuth, and Open ID Connect)	`components`
Security Requirements Object	Lists the security schemes required for a given operation (if in `path`) or required globally (if in `security` off the Open API Object)	`paths` -> `path item` -> `operation` -> `security` or `security`
OAuth Flows Object and OAuth Flow Object	Provides configuration details for OAuth	`component` -> `securitySchemes` -> `OAuth Flows Object` -> `OAuth Flow Object`

3.3.1 Define the Security Scheme Object

Based on analysis from Sections 3.1 and 3.2, project teams **must** define the security schemes used by the API. This Security Scheme Object defines a security scheme that can be used by operations. The security scheme **must** be defined using an available technique (e.g., HTTP authentication, API key, OAuth2, or OpenID Connect Discovery).

When the scheme is applied to an individual operation, that operation **must** also reference the corresponding security scheme. Alternatively, the scheme **must** be applied globally across the API.

- Define the `securitySchemes` object within the `components` object, based on the OpenAPI specification.

[44] For example, numerous tools are listed on the site at https://openapi.tools/.

- Determine security requirements and define the needed corresponding types (e.g., Hypertext Transfer Protocol (`http`), `apiKey`, `oauth2`, and `openIdConnect)` in the `securitySchemes`.[45] These four are the only supported security schemes in Open API version 3.0.
- Figure 1 provides a brief example of defining an `apiKey` type in Open API format.

```
{
   "components": {
      "securitySchemes": {
         "api_key": {
         "type": "apiKey",
         "name": "api_key",
         "in": "header"
         }
      }
   }
}
```

Figure 1: Example of securitySchemes in OpenAPI 3.0 defining UserAccess

- This specifies the scheme for prominent security features used by the API (e.g., API key, authorization technique).[46] HTTP Basic, HTTP Bearer, and API keys only provide a measure of confidentiality and should be used with HTTPS.
- APIs that are not processing open data should implement authorization.
- In OAuth2,[47] authentication occurs separately as part of the user session in OAuth.

• Define the scope of the `securitySchemes` for the API when using OAuth 2 or Open ID Connect.[48]
  - The scope can apply to the entire API or to individual operations. When the scope is placed in the `security` section on the root, it applies to the entire API. When the scope is provided within an operation, the applicability is limited to that

[45] The VA TRM restricts usage of Open ID Connect. Refer to https://www.oit.va.gov/Services/TRM/StandardPage.aspx?tid=8252 and https://www.oit.va.gov/Services/TRM/StandardPage.aspx?tid=6769.
[46] Refer to the OpenAPI 3.0 Specification, Security Definitions Object, at https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#securityDefinitionsObject.
[47] Refer to OAuth2 information in the OAuth 2.0 Security Primer, OAuth 2.0 Implicit Grant, and OAuth 2.0 Authorization Code Grant Electronic Data Patters (EDPs). Some APIs, such as those communicating open data, should be open to the public and may not implement this security control, based on OMB Memo 13-13 at https://project-open-data.cio.gov/policy-memo/.
[48] HTTP authentication (e.g., HTTP Basic, HTTP Bearer) and API keys do not use scopes.

operation.[49] For OAuth2, the scopes used in security **must** be defined in `securitySchemes`.[50]

- Use cases should be considered when determining security for OpenAPI scope statements.
- Authentication **must** be defined either globally and/or locally for each resource to avoid security gaps.
- To determine the proper VA enterprise connections, research services from the VA API Developer Portal and the VA Office of Identity, Credential, and Access Management (OICAM) information at http://vaww.oicam.va.gov/.[51]
- The following example shows a Security Scheme object for OAuth2:

```
{
   "type": "oauth2",
   "flows": {
      "implicit": {
         "authorizationUrl": "https://example.va.gov/authorizationservice",
         "scopes": {
            "write:obj": "Write object from example app",
            "read:obj": "Read object from example app"
         }
      }
   }
}
```

- For example, to apply OAuth2 to the `account_info` GET operation, the following security information could be applied. If a scope is used, it **must** be previously defined in the `securitySchemes` object.

```
{
   "paths": {
      "account_info": {
         "get": {
            "security": {
            "OAuth2": ["write:obj", "read:obj"]
            }
         }
      }
   }
```

[49] The Operation Object is found within the Path Item Object, which is found inside the Paths Object.
[50] For example, the VA API Platform OpenID Connect feature includes existing scopes such as profile, service history.view, disability_rating.view, and veteran_status.view. For more information, refer to http://developer.va.gov/oauth#scopes.
[51] Also, the Access Services (AcS) playbook is available at https://vaww.oed.portal.va.gov/sites/vrm/IAM/playbooks/Pages/AcS%20Playbook%20Home.aspx; IAM services information and URLs can be found at https://vaww.oed.portal.va.gov/sites/vrm/IAM/playbooks/Pages/IAM%20URLs.aspx.

```
}
```

3.3.2 Define the OAuth Flows Object When Using OAuth or Open ID Connect

This object is used to set the configuration of the supported OAuth flows (i.e., grant types) including the `implicit` flow, the `password` flow, the `clientCredentials` flow, and the `authorizationCode` flow. Each of these flow types has a separate OAuth Flow Object (a subset of the OAuth Flows Object) to define it. The following shows an OAuth implicit flow example for a hypothetical system. Other flow types will require different fields to be defined.

```
{
   "type": "oauth2",
   "flows": {
      "implicit": {
         "authorizationUrl": "https://example.va.gov/api/oauth",
         "scopes": {
            "write:obj": "write new record for Example system",
            "read:obj": "read operation for Example system"
         }
      }
   }
}
```

3.3.3 Define the Security Requirement Object

The project team documenting the API can define a Security Requirement Object to specify the required security schemes for either an individual operation (i.e., where this is defined in `security` in the Operation Object) or for the API in its entirety (i.e., where this is defined in `security` in the overall OpenAPI Object). These should be set using the principle of least privilege so that access and information is granted only for necessary, legitimate use cases.

The field pattern used (e.g., ‘api_key’, ‘bearerAuth’, etc.) **must** match a corresponding security scheme defined in the Security Scheme object, within the Components object. In the case of HTTP authentication and API key security schemes, the string **must** be empty (i.e., the scope brackets are empty and written as “[]”). Note the following API key example for reference.

```
{
   "api_key": []
}
```

For ‘oauth2’ and ‘openIdConnect’ schemes, the string **must** contain a list of scopes. In the following example, these are the scopes “write:obj” and “read:obj” as defined in the brackets.

```
{
   “example”: [
      “write:obj”,
      “read:obj”
   ]
}
```

3.4 Address Additional OpenAPI Security and Related Settings

Additional security conditions include error responses and Cross-Origin Resource Sharing (CORS). The following subsections detail how to set these.

3.4.1 Set Additional Fields for Security

- Determine and define Open API responses for the production API and service. This **must** include errors with fields and information that are limited and prevent data leaks.[52] In the production version of the API, the API responses **must** not include stack traces or database output. The responses shall be separate from the service's full error logging, but can include information that can be used to reference the full error logging. This error content **must** be documented within the Operation Object, which is within a Path Item Object, inside of a Paths Object.
- When a request is not authorized, the API and service should return the authorization error (HTTP status code 401), instead of the *file not found* error (HTTP status code 404), to prevent an unauthorized mapping of the file and directory structure.
- Set any applicable rate limit information in the Response Object, which is inside the Operation Object. The Operation Object is part of the Path Item Object, which is part of the Paths Object. This should be set so that a request from a single source does not limit the availability of the service. The following is an example of the rate limit, defined inside the Response Object.

```
{
   "headers": {
      "X-Rate-Limit-Limit": {
         "description": "The number of allowed requests in a time period",
         "schema": {
            "type": "integer"
         }
      }
   }
}
```

3.4.2 Configure Cross Origin Resource Sharing (CORS)

CORS uses HTTP headers to communicate with a user client or end point. These headers inform a user client whether applications running on one domain can also allow access to resources on a different domain. Project teams should follow these points:

- If multiple origins are not required for the API solution, disable CORS.

[52] For additional information, refer to the Open Web Application Security Project (OWASP) at https://www.owasp.org/index.php/REST_Security_Cheat_Sheet.

- However, if multiple origins are needed, these origins should be explicitly defined, specified, and allowed. When used, CORS needs to be enabled on the corresponding web server or API gateway. HTTP headers should include the `Access-Control-Allow-Origin` parameter set so that multiple origins are permitted. If used, client users will need to access the API and service through web browsers, or other software that supports CORS.

4 Application

The security compliance user stories have a standard for service access and documentation, VistA Integration Control Registrations (ICRs), web traffic, and performance analytics. Project teams using the VIP **must** comply with the VA TRM; and map to the security compliance user stories below. Future changes in the standard will be reflected in the VA TRM; and in pertinent security compliance user stories that are related to both API consumption and provisioning.[53]

Table 4: SEC User Stories

Security User Story	Title	User Story Text[54]	Relevant User Story Acceptance Criterion
SEC.01.06.01	Security	As the official that ensures all forms of protected health information (PHI) and personally identifiable information (PII) that is collected, used, and maintained according to the Health Insurance Portability and Accountability Act (HIPAA) and VA mandates, I want to certify that the processes used are completed; so that they are in accordance with HIPAA, VA Directive 6066, VA 6500, VHA Handbook 1605.05, and are easily understood by all.	1. Ensure the following elements are found in the Privacy Impact Assessment (PIA): a. The name and acronym for each system identified b. The types of PII contained in that system; c. Classification of level of sensitivity of all types of PII, as combined in that information system (IS) d. Classification of level of potential risk of substantial harm, embarrassment, inconvenience, or unfairness to affected individuals, as well as the financial or reputational risks to VA.

[53] Refer to the Open API Specification at the TRM at https://www.oit.va.gov/Services/TRM/StandardPage.aspx?tid=8259.

[54] The table references the official SEC user stories that are published and used by the VA Strategic Technology Alignment Team (STAT) Team.

Security User Story	Title	User Story Text[54]	Relevant User Story Acceptance Criterion
			2. Establish, maintain, and update annually an inventory that contains a listing of all programs and ISs identified as collecting, using, maintaining, or sharing PII. 3. Provide each update of the PII inventory to the chief information officer (CIOI and information security officer (ISO) annually to support the establishment of information security requirements for all new or modified ISs containing PII. 4. Develop, implement and maintain a Privacy Incident Response Plan, which may or may not be included within the Security Incident Response Plan(s), which provide an organized and effective response to privacy incidents.
SEC.02.07.02	Identification and Authentication	As an authorizing official and/or system owner, I want to verify that the VA systems/network requires organizational users to uniquely identify and authenticate into company information assets using password, tokens, biometrics, multi-factor authentication, or some combination thereof so that there is not an issue of improper use and handling of sensitive information.	1. (S) Ensure the IS uniquely identifies and authenticates organizational users (or processes acting on behalf of organizational users). (IA-2) 2. (S) Ensure the IS implements multifactor authentication for network access to privileged and non-privileged accounts. (IA-2 [1,2]) 3. (S) Ensure the IS implements multifactor authentication for local access to privileged and non-privileged accounts. (IA-2 [3,4]) 4. (S) Ensure the IS Owner implements replay-resistant authentication

Security User Story	Title	User Story Text[54]	Relevant User Story Acceptance Criterion
			mechanisms for network access to privileged accounts. (IA-2 [8,9]) 5. (S) Ensure the IS implements multifactor authentication for remote access to privileged and non-privileged accounts, such that one of the factors is provided by a device separate from the system gaining access, and the device meets the strength of mechanism requirements. (IA-2 [11]) 6. (S) Ensure the IS accepts and electronically verifies personal identity verification (PIV) credentials. (IA-2 [12])

5 Impacts

If standard API documentation approaches are not followed, the following negative impacts may be realized across the VA enterprise and projects:

- If security is not properly documented for APIs, API consumers and users may not properly configure and securely use the API, resulting in security gaps.
- If projects do not plan for security when implementing documentation, secure features are unlikely to be integrated into the API and corresponding service.

Appendix A: References

- 18F GSA API Standards: https://github.com/18F/api-standards
- Access Services (AcS) playbook, https://vaww.oed.portal.va.gov/sites/vrm/IAM/playbooks/Pages/AcS%20Playbook%20Home.aspx
- API Documentation Standard, June 2018: https://www.oit.va.gov/library/files/edp/apis/APIEDP_DocumentationStandard_v1.pdf
- APIs for Data Security and Privacy Part One - Apigee's testimony before the ONC API Task Force. Greg Brail.
- APIs: A Strategy Guide. Daniel Jacobson, Greg Brail, Dan Woods.
- Best Practices for Building Secure APIs. Rakesh Talanki and Vidheer Gadikota. https://medium.com/apis-and-digital-transformation/best-practices-for-building-secure-apis-2b4eb8071d41
- FHIR Release 3, FHIR Security, see https://www.hl7.org/fhir/security.html
- FHIR Release 3, RESTful API, see https://www.hl7.org/fhir/http.html
- FHIR Release 3, Security and Privacy Module, see https://www.hl7.org/fhir/secpriv-module.html
- IAM services information and URLs, https://vaww.oed.portal.va.gov/sites/vrm/IAM/playbooks/Pages/IAM%20URLs.aspx
- IETF Hypertext Transfer Protocol (HTTP/1.1) Authentication Proposed Standard, https://tools.ietf.org/html/rfc7235
- NIST Special Publication 800-175B. Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms. http://dx.doi.org/10.6028/NIST.SP.800-175B
- Office of Management and Budget Memorandum 13-13: https://digital.gov/open-data-policy-m-13-13/
- Office of Management and Budget Memorandum 17-06: https://www.whitehouse.gov/sites/whitehouse.gov/files/omb/memoranda/2017/m-17-06.pdf
- Open Web Application Security Project (OWASP): https://www.owasp.org/index.php/REST_Security_Cheat_Sheet
- OpenAPI 3.0 Specification, Security Scheme Object: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.2.md#securitySchemeObject
- OpenAPI Specification, https://github.com/OAI/OpenAPI-Specification
- OpenAPI Tools, https://openapi.tools/
- STRIDE Threat Model: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)
- VA Directive 6551: https://www.va.gov/vapubs/viewPublication.asp?Pub_ID=829&FType=2
- VA EDPs: https://www.oit.va.gov/library/recurring/edp/
- VA Open Data Portal: https://www.data.va.gov/

- VA SEC User Stories: https://vaww.portal2.va.gov/sites/asd/AERB/FISMASecurityCompliance/SitePages/Home.aspx
- VA TRM: https://www.oit.va.gov/Services/TRM/StandardPage.aspx?tid=8252 and https://www.oit.va.gov/Services/TRM/StandardPage.aspx?tid=6769
- VIP Guide version 3.2: https://vaww.vaco.portal.va.gov/sites/OIT/epmo/vip/Policy%20%20Guidance/VIP%20Guide%203.2.pdf.
- Reference Open API 3.0 example, VA Facilities API, https://dev-api.va.gov/services/va_facilities/docs/v0/api