



API Security Standard

DOC-IT-IS-24-1-3

Revision 2.2

APRIL 10, 2026

This document contains proprietary information, which is the exclusive property of RXO, Inc. and may not be used or reproduced, in whole or in part, without the express permission of RXO, Inc.

Copyright © 2026 RXO, Inc. All rights reserved.

REVISION HISTORY

This revision history lists all changes to this document, the date of each revision, and the reason for each revision.

<u>Rev.</u>	<u>Rev. Date</u>	<u>Revision Description</u>
1.0	11/01/2022	<i>Initial Version</i>
2.0	11/01/2024	<i>Annual Review</i>
2.1	11/01/2025	<i>RXO, Inc. Revisions, annual review</i>
2.2	4/10/2026	Added new sections on sanctioned tooling controls, API lifecycle governance, credential handling, and third-party API consumption

Table of Contents

1 Executive Summary	4
2 Purpose	4
3 Scope	4
4 API Standard	5
4.1 Asset Management	5
4.2 Access Control	5
4.3 Authentication	6
4.4 Encryption	6
4.5 Data Restriction & Validation	7
4.6 Load Balancing & Throttling	7
4.7 API Service Registry Configuration	7
4.8 API Service Mesh Configuration	8
4.9 API Gateway Configuration	8
4.10 Audit & Logs	9
4.11 Threat Detection, Monitoring & Analysis	10
4.12 Sanctioned Tooling & Account Controls	10
4.13 API Lifecycle Governance	10
4.14 Third-Party API Consumption	11
5 Roles & Responsibilities	11
5.1 Application Owners	11
5.2 Network Management Team	12
5.3 InfoSec Team	12
6 Compliance	13
6.1 Compliance Measurement	13
6.2 Non-Compliance	13
7 Exceptions	13
8 Review & Updates	13
9 References	14
9.1 Related Company Standards & Policies	14
9.2 Related Industry Best Practices, Standards & Policies	14
10 Definitions	15

1 Executive Summary

A foundational element of innovation in today's app-driven world is the API, or Application Programming Interface. From banks, retail, and transportation to IoT, autonomous vehicles and smart cities, APIs are a critical part of modern mobile, SaaS and web applications and can be found in customer-facing, partner-facing and internal applications. Many RXO applications use internal and/or external APIs.

By nature, APIs expose application logic and sensitive data such as Personally Identifiable Information (PII) and therefore have increasingly become targets for attackers. Without secure APIs, rapid innovation would be impossible.

API Security focuses on strategies and solutions to understand and mitigate the unique vulnerabilities and security risks of APIs.

2 Purpose

The purpose of this document is to define the API security standards within RXO. API security standards are defined to identify potential or realized weaknesses due to inadvertent misconfiguration, weak authentication, insufficient error handling, sensitive information leakage, etc.

Prevention, detection, and subsequent mitigation of these issues will limit the attack surface of RXO API services available both internally and externally, protect customers' sensitive data, and satisfy compliance with any relevant policies in place.

3 Scope

This standard applies to RXO, Inc., including all subsidiaries, divisions, and operating entities. All directors, officers, employees, contractors, and third parties acting on behalf of RXO must comply with this standard.

This standard covers all internal and external APIs used by any group, department, or application developed or used by RXO, including all of its business units, to store, process or otherwise manage customer data or sensitive company data.

All application owners, including those of cloud-based or third-party applications, are responsible for taking the appropriate steps to secure their APIs.

Systems or applications that cannot comply with these standards require an exception from the Infosec team.

4 API Standard

4.1 Asset Management

- 4.1.1 All APIs, including version control information, must be fully documented and added to the company Asset Inventory, including internal and external APIs or APIs used by cloud-based systems. The inventory must include A. The API identification, B. Describe the caller and intended/expected callee, Equivalence classes between URLs (as per proxy arrangements, round-robin, failover, etc.) must be denoted. C. default contracts, D. Data classification of inbound requests and outbound responses.
- 4.1.2 API owners are responsible for maintaining the accuracy of their inventory entries throughout the API's operational life. Stale or inaccurate inventory entries must be corrected within 30 days of a change to the API's scope, ownership, or environment.
- 4.1.3 Deprecated or retired API versions must be formally decommissioned—not merely undocumented. Decommissioning must be reflected in the API inventory and confirmed by IT Operations. A defined deprecation notice period must be communicated to all known consumers prior to retirement.
- 4.1.4 Legacy or undiscovered APIs (Shadow APIs) identified through security assessments or discovery scanning must be inventoried, assessed for risk, and either formally documented and secured or decommissioned on a timeline approved by InfoSec.
 - 4.1.4.1 All relevant hosts should be updated in the Asset Inventory.
- 4.1.5 The company Asset Management Policy will apply to all APIs, including internal and external APIs or APIs used by cloud-based systems.

4.2 Access Control

- 4.2.1 The company Access Control Policy will apply to all APIs, including internal and external APIs or APIs used by cloud-based systems.
- 4.2.2 Access policies to all APIs and their resources should be defined and provisioned to an access server.
- 4.2.3 Access control policies must separate Administrative functions from User functions.
- 4.2.4 The access server should be capable of supporting fine-grained policies.
- 4.2.5 Access policies at a coarse level of granularity, say “Permit to Call for a given set of addressable functionalities”, should be defined and enforced at the initial API gateway (see API Gateway Section below).
- 4.2.6 Authorizations at the finer level of granularity (e.g., related to domain of the microservices' business logic) should be defined and enforced closer to the location of the microservices (e.g., at the micro-gateway) or sometimes at the microservice itself.
- 4.2.7 Object level authorization checks should be considered in every function that accesses a data source using an input from the user.
- 4.2.8 Access decisions from the access server should be conveyed to individual and sets of microservices through standardized tokens encoded in a platform-neutral format (e.g., OAuth 2.0 token encoded in JSON format). The token can be either a handle-based token or an assertion bearing token.

- 4.2.9 The scope of internal authorization tokens appended by the micro gateway or decision point to each request should be carefully controlled; for example, in a request for transaction, the internal authorization token should be limited in scope to only involve the API endpoints that must be accessed for that transaction.

4.3 Authentication

- 4.3.1 Authentication to APIs that have access to sensitive data should not be done simply by using API keys. Access to such APIs should require authentication tokens that have either been digitally signed (e.g., client credentials grant) or that are verified with an authoritative source.
 - 4.3.1.1 Additionally, some services may require either single-use tokens or short-lived tokens (tokens that expire after a short time-period) to limit the damage a compromised token can cause.
- 4.3.2 Authentication tokens should be used for access, and should be handle-based, cryptographically signed, or protected by an HMAC scheme.
- 4.3.3 Every API Key that is used in the application should have restrictions specified both for the applications (e.g., mobile app, IP address) and the set of APIs where they can be used.
- 4.3.4 Internal authorization tokens must not be provided back to the user, and the user's session tokens must not be passed beyond the gateway for use in policy decisions.
- 4.3.5 The restriction scope for functionality of every API Key should be commensurate with the level of assurance provided during identity proofing, whether it be machine or human driven identity proofing.
- 4.3.6 When stateless authentication tokens (e.g., JSON Web Tokens (JWT)) are used by implementing shared libraries associated with a microservice, the following security precautions must be observed:
 - 4.3.6.1 The token expiry times should be as short as possible since they determine the duration of the session, and an active session cannot be revoked.
 - 4.3.6.2 The token secret key must not be a part of the library code, it must be a dynamic variable represented by an environmental variable or specified in an environment data file.

The key value should be stored in a data vault solution.
- 4.3.7 Use of a run-time prevention strategy for credential abuse is preferable to an offline strategy.
- 4.3.8 Establish a threshold for a designated time interval from a given location (e.g., IP address) for the maximum number of login attempts; if the threshold is exceeded, preventive and alerting measures must be triggered by the authentication/authorization server.

4.4 Encryption

- 4.4.1 Encryption must be used for all API communications.

- 4.4.2 The latest Transport Layer Security (TLS) or Web Security Services (WSS) versions must be used to avoid the use of the weakest cipher suites.
 - 4.4.2.1 SOAP messages, with the built-in WSS protocol, should be considered for the most secure or sensitive APIs.
 - 4.4.2.2 REST implementation, using TLS and JSON files, which is generally faster than SOAP messaging, may be used for less sensitive applications.

4.5 Data Restriction & Validation

- 4.5.1 Data and object properties used or revealed in API requests or responses should be kept to a minimum, especially in error messages. Adding additional information to reduce potential future coding changes is discouraged.
- 4.5.2 No sensitive data, as defined in the Company Information Classification Standard, should be used or exposed in an API request or response unless absolutely required to perform the functions of the API.
- 4.5.3 Ensure that no IP addresses are used or revealed in API requests, responses, or error messages.
- 4.5.4 Verbose error messages should not be exposed to external users.
- 4.5.5 API data accepted in a request must be inspected to reject embedded content or extensive data. Use JSON or XML schema validation to avoid any SQL injection or XML bombs and check if any expected parameters are as they should be.

4.6 Load Balancing & Throttling

- 4.6.1 Load balancing and throttling (rate limiting) should be used to manage API requests.
- 4.6.2 The load balancing function should be decoupled from individual service requests; for example, the program that performs health checks on the services to determine the load balancing pool should run asynchronously in the background.
- 4.6.3 Limits should be implemented and determined based on well-defined, documented API usage plans.
- 4.6.4 Quotas or limits for application usage should be based on both infrastructure and application-related requirements.
- 4.6.5 For high security APIs, such as those using sensitive data, replay detection must be implemented. Based on the risk, this feature can be configured to detect replays 100% of the time (sensitive data) or perform random detection (for less sensitive data).

4.7 API Service Registry Configuration

- 4.7.1 Service registry capabilities should be provided through servers that are either dedicated or part of a service mesh architecture.
- 4.7.2 Service registry services should be in a network that has been configured with certain QoS parameters to ensure its availability and resilience.

- 4.7.3 Communication between an application service and a service registry should be through a secure communication protocol, such as HTTPS/TLS.
- 4.7.4 Service registry should have validation checks to ensure that only legitimate services are performing the registration and refresh operations or querying its database to discover services.
- 4.7.5 The bounded context and loose coupling principle for microservices should be observed for the service registration / deregistration function. See NIST SP 800-204 for additional details.
- 4.7.6 Service self-registration / deregistration patterns should be avoided.
- 4.7.7 Registration or deregistration of an application service should be enabled using a third-party registration pattern.
 - 4.7.7.1 The application service should be restricted to simply querying the service registry for service location information as described in the client-side discovery pattern.
 - 4.7.7.2 If a third-party registration pattern is implemented, registration / deregistration should only take place after performing a health check on the application service.
 - 4.7.7.3 Distributed service registry should be deployed for large microservices applications, and care should be taken to maintain data consistency among multiple service registry instances.

4.8 API Service Mesh Configuration

- 4.8.1 Policy support should be enabled for:
 - 4.8.1.1 Designating a specific communication protocol between pairs of services.
 - 4.8.1.2 Specifying the traffic load between pairs of services based on application requirements.
- 4.8.2 The default configuration should always be to enable access control policies for all services.
- 4.8.3 Avoid configurations that may lead to privilege escalation (e.g., the service role permissions and binding of the service role to service user accounts).
- 4.8.4 Service mesh deployments should have configuration capabilities to specify resource usage limits for its components. The absence of this feature creates the potential for these components to impact the resiliency and availability of the overall microservices application.
- 4.8.5 Service mesh deployments should have configuration capabilities to collect and send environment metrics, including request metrics, to the centralized service for monitoring.
- 4.8.6 Policies should allow for specifying either a single service mesh or multiple service meshes (each with their own control plane) for multi-cluster microservices environments to ensure high availability and resiliency.

4.9 API Gateway Configuration

- 4.9.1 Clients should not be configured to call their target services directly but rather be configured to point to a single gateway URL – an “API gateway”.

- 4.9.2 Whenever possible, use IP Whitelists and IP Blacklists at the gateway to restrict access to APIs and their resources.
- 4.9.3 The API gateway should be leveraged to centralize enforcement of authentication and access control for all downstream APIs, eliminating the need to provide authentication and access control for each of the individual services.
 - 4.9.3.1 Mitigating controls such as mutual authentication should be leveraged to prevent direct, anonymous connections to the services.
 - 4.9.3.2 Client-to-API-gateway as well as Service-to-Service communication should take place after mutual authentication.
- 4.9.4 Integrate the API gateway with an identity management application to provision credentials before activating the API.
- 4.9.5 When identity management is invoked through the API gateway, connectors should be provided for integrating with identity providers (IdPs).
- 4.9.6 The API gateway should have a connector to an artifact that can generate an access token for the client request (e.g., OAuth 2.0 Authorization Server).
- 4.9.7 Distributed gateway deployments, or a combination of initial gateway that intercepts all client accesses, and micro-gateways closer to micro-services should have a token translation (exchange) service between gateways.
 - 4.9.7.1 The token presented to the initial gateway should have permissions with a broad scope whereas the token presented to inside gateways (or micro-gateways) should be more narrowly scoped with specific permissions or an entirely different token type that is appropriate for the target micro-service platform. This helps to implement the least privilege paradigm.

4.10 Audit & Logs

- 4.10.1 API traffic metadata must be collected and sent to the central logging system or security information and event management (SIEM) system.
- 4.10.2 At a minimum, traffic should be logged across the APIs:
 - 4.10.2.1 Per user
 - 4.10.2.2 Per IP
 - 4.10.2.3 Per token
 - 4.10.2.4 Per API.
- 4.10.3 Input validation errors and extra parameters errors, crashes and core dumps must be logged.
 - 4.10.3.1 A class of software that can accomplish this is the OWASP “AppSensor” which could potentially be implemented in the gateway, service mesh and micro-service itself.

4.11 Threat Detection, Monitoring & Analysis

- 4.11.1 Securely channel all traffic information to a monitoring and/or analytics application for detecting attacks (e.g., denial of service, malicious actions) and unearthing explanations for degrading performance.
- 4.11.2 Security monitoring should be performed at both the gateway and service level to detect, alert and respond to inappropriate behavior, for example a bearer token reuse attack and injection attacks.
- 4.11.3 Configure IDS and boundary devices to detect the following at the gateway:
 - 4.11.3.1 A denial-of-service attack and raise an alert before the service is no longer accessible.
 - 4.11.3.2 A distributed network probe.
- 4.11.4 Configure service hosts to scan file uploads and the contents of each container's memory and file system for resident malware threats.
- 4.11.5 Continuous runtime monitoring should be conducted to discover errors before they become incidents.
- 4.11.6 If possible, use a traffic dashboard specific to APIs and the factors noted in the Audit & Logs section above.
- 4.11.7 The Company Security Incident Response procedures should be updated to include API security incidents, indicators, and responses.

4.12 Sanctioned Tooling & Account Controls

- 4.12.1 Only company-approved and IT-managed tools are authorized for API development, testing, documentation, and exploration. The use of unapproved personal tools or unsanctioned SaaS platforms for these activities is prohibited.
- 4.12.2 All access to approved API tooling (including, but not limited to, Postman and equivalent platforms) must be provisioned through corporate-managed accounts using RXO's official email domain (@rxo.com). Personal accounts, free community-tier accounts, and any account linked to a personal email address are strictly prohibited for any RXO business purpose.
- 4.12.3 Any workspace, collection, project, or documentation repository within a third-party API tool that contains RXO API information must be configured as Private. The creation or use of public workspaces or publicly accessible repositories containing RXO API data constitutes a policy violation.
- 4.12.4 The synchronization or persistent storage of RXO API metadata, endpoint definitions, or example payloads in unapproved third-party SaaS environments is prohibited regardless of the sensitivity level of the data involved.
- 4.12.5 API specifications must be produced, version-controlled, and submitted to InfoSec as part of the design review process. Specifications must not be publicly accessible unless the API itself is classified as Public.

4.13 API Lifecycle Governance

- 4.13.1 All APIs must be registered in RXO's API inventory (Section 4.1) prior to promotion beyond testing environments. Registration is a prerequisite for any environment promotion.

4.13.2 The RXO InfoSec team must be engaged during the Design Phase of any new API and for material changes to existing APIs that expand scope, change authentication model, or alter data exposure. This engagement must occur before code is promoted beyond a development environment.

4.13.3 Security requirements identified during the design phase must be addressed before any API is promoted to production. No API may enter production with unresolved Critical or High severity security findings unless a formal exception has been approved.

4.13.4 All API releases must pass through RXO's change control process. InfoSec security assessments are a required gate within that process.

4.13.5 Credentials used in development and testing environments must be separate from production credentials. Cross-environment credential reuse is prohibited.

4.13.6 Production credentials, API keys, client secrets, and bearer tokens must never be hardcoded in source code, committed to version control systems, or synchronized to any unapproved cloud-based testing or collaboration tool.

4.13.7 In the event of a known or suspected credential compromise, the relevant API owner must immediately initiate the emergency revocation process and notify the InfoSec team. Delay in reporting a credential compromise is itself a policy violation.

4.14 Third-Party API Consumption

4.14.1 All third-party APIs consumed by RXO applications must be assessed for security posture and approved by InfoSec prior to production integration. Assessments must consider the vendor's authentication model, data handling practices, and security disclosure history.

4.14.2 Vendor contracts and agreements involving API integrations must include appropriate security obligations, breach notification requirements, and RXO's right to request security attestation or conduct audits. Procurement and Legal must be engaged for new third-party API integrations.

4.14.3 Credentials issued by external API providers for use in RXO systems are subject to the same handling requirements as RXO-issued credentials under Section 4.13.5–4.13.7.

4.14.4 Third-party APIs must be inventoried in RXO's API inventory in accordance with Section 4.1.

5 Roles & Responsibilities

5.1 Application Owners

- 5.1.1 Application owners must document their APIs and submit them to the InfoSec team for review and approval prior to implementation.
- 5.1.2 Documentation should also be provided to the Incident Response Team to ensure they are aware of any new potential threat surfaces.
- 5.1.3 Application owners must notify the Network Management Team of any new APIs or API partners so that the API Gateway may be configured as needed to support the new APIs or new API partners.

- 5.1.4 Application owners must ensure logging information is sent to the central logging system to meet all requirements of the Audits & Logs section above and of the Company Information Logging & Monitoring Standard.

5.2 Network Management Team

- 5.2.1 The Network Management Team (NMT) will manage the API Gateway.
- 5.2.2 NMT will configure and secure ports and services needed at the API Gateway.
- 5.2.3 NMT will apply IP Blacklist and IP Whitelists as appropriate at the API Gateway.

5.3 InfoSec Team

- 5.3.1 The InfoSec team will conduct security assessments of any new or updated APIs.

6 Compliance

6.1 Compliance Measurement

The InfoSec team will verify compliance to this standard through various methods, including but not limited to, periodic walk-throughs and interviews, business tool reports, internal and external audits, and feedback to the standards owner(s).

6.2 Non-Compliance

An employee found to have violated this standard may be subject to disciplinary action, up to and including termination of employment.

API security assessments are a requirement of the change control process and API owners are required to adhere to this standard unless found to be exempt. All API releases must pass through the change control process. Any APIs that do not adhere to this standard may be taken offline until such time that a formal assessment can be performed at the discretion of the Chief Information Officer.

7 Exceptions

Any exceptions to this standard must be formally documented and approved in writing by the RXO Governance Risk and Compliance (GRC) team. Exceptions documentation must include a valid business justification and compensating controls. Exceptions are time-bound and subject to periodic review.

8 Review & Updates

This standard will be reviewed annually, in response to cryptographic vulnerabilities, upon major infrastructure or threat landscape changes, or as required due to changes in business, technical, or regulatory requirements.

9 References

Detailed processes and procedures must be written to implement the standards in this document.

9.1 Related Company Standards & Policies

These policies and standards can be found on the GSO Security Policies and Standards SharePoint site.

- 9.1.1 Asset Management Policy
- 9.1.2 Access Control Policy
- 9.1.3 Information Classification Standard
- 9.1.4 Security Incident Response Policy
- 9.1.5 Information Logging & Monitoring Standard

9.2 Related Industry Best Practices, Standards & Policies

- 9.2.1 [OWASP API Security Top 10](#)
- 9.2.2 [NIST SP 800-204 - Security Strategies for APIs](#)

10 Definitions

API	Application Programming Interface
Micro-services	An architectural approach to building applications where each core function, or service, is built and deployed independently.
OAuth	Open Authentication, an open standard authentication protocol used for passing authorization from one service to another without sharing actual user credentials.
JSON	JavaScript Object Notation, an easily human readable data-interchange format.
HMAC	Hash-based Message Authentication Code.
TLS	Transport Layer Security, a cryptographic protocol designed to provide communications security over a computer network. The protocol is widely used in applications such as email, instant messaging, and voice over IP, but its use in securing HTTPS remains the most publicly visible.
WSS	Web Security Services, mechanisms to provide protection through message integrity, and single message authentication; built-in protocol used by SOAP APIs.
SOAP	Simple Object Access Protocol, a secure, strongly typed messaging framework that relies heavily on XML and schemas.
REST	Representational State Transfer, a common API implementation that supports TLS.
XML	Extensible Markup Language, a simple text-based format for representing structured information.
QoS	Quality of Service
GSO	RXO's Global Security Office
OWASP	Open Web Application Security Project
NIST	National Institute of Standards

Certificate of Review

API Security Standard

July 2nd 2026 2:10:21 pm

Name	Job title	Stakeholder type	Action	Timestamp
Mark Gutman		Policy Owner	Submitted	Wednesday, June 17, 2026 9:15 AM EDT
Faisal Khan		Policy Approver	Approved	Tuesday, June 30, 2026 12:27 PM EDT
Mark Gutman		Policy Admin	Published	Thursday, July 2, 2026 2:10 PM EDT