API 標準設計ガイド・基礎編
（付録：サイト利用規約テンプレート）

独立行政法人
情報処理推進機構

デジタル基盤センター
デジタルエンジニアリング部 データスペースグループ

変更履歴

変更日	変更内容
2025年3月26日	初版発行

目次

本資料の概要

本資料作成の背景

欧州では 2018 年施行の「第 2 次決済サービス指令（PSD2: Payment Services Directive 2）」【1】に基づき、金融業界において、顧客の同意の基で第三者サービスプロバイダーに対し、API（アプリケーション・プログラミング・インターフェース）を通じた金融データへのアクセス提供が義務化された。これにより欧州、米国の金融業界では、参照系・更新系の銀行 API の標準化が進み、BIAN(金融業界の相互運用性促進のために設立された国際組織)【2】により、2021/12 時点で 243 個の API が提供された。（「オープン API をめぐるわが国の現状と展望」（デジタル庁 2023 年）【3】より引用）

このことから欧州では官民が協力して API の構築と利用を戦略的に推進していると考えられる。（考察①）

表 1：日米のマイクロサービス/API の導入状況　（単位：%）

調査年	調査対象	導入済	検討中	導入未定
2021 年	日本	20.1	14.8	65.1
	米国	51	23.3	25.7
2024 年	日本	21	16.1	62.9
	米国	57.5	17.4	25.1

「DX 白書 2021」「DX 白書 2023」【4】（IPA）を基に作成

一方日本の API 導入状況について、IPA の「DX 白書 2021」と「DX 白書 2023」から日米での API 導入状況を比較する。

- ✓ 米国の API 導入済は 2021 年から 2024 年の 3 年間で約 7%伸びたが日本の API 導入済は 3 年間ほぼ横ばいである。
- ✓ 3 年間で API 導入済の日米差は約 7%に拡大した。

このことから日本は米国に比較すると API の導入済の比率が低く、導入率向上の傾向が弱いことが確認できる。（考察②）

上記考察①②より、日本では官民が一体となって API の導入を推進することが求められる。また API の導入が進めば、業界内、組織間のデータ連携が促進されると考えられる。

本資料の目的

目的１

本資料では、API 標準設計書の読み方と書き方の基礎を紹介する。日本では、データ連携時に API を活用せず、データ利用者が個別に設計・開発するケースが多い。この状況を踏まえ、本資料では、API 標準設計書の普及により API の利活用が進み、API 提供及び API 利用の開発に関わる作業が効率化されることを目的とする。

目的２

API やデータを自組織のサイトで提供する場合、権利保護や想定外の利用を抑止するため、利用に関して、API、データ、個人情報保護に関する規約を定義する必要がある。本資料では政府標準を参考にしたサイト利用規約のテンプレート（以下、「利用規約のひな型」と表記）を紹介する。利用規約のひな型を活用すれば API やデータを自組織のサイトで提供する際に、関連法規の調査や法律の専門家への問い合わせを減らすことができる。また、自組織に合わせたカスタマイズも可能であるため、官民を問わず、効率的かつ柔軟に利用規約を作成できる。

API 標準設計と利用規約のひな型を活用することで、現場の技術者から所属組織の経営層や技術部門へ、API や利用規約のひな型等の標準を導入することによる効率化のメリットが伝わり、DX やデータ連携の推進が期待される。

本資料の想定読者

本資料は、API の提供者、API 利用者、及び API・データ・個人情報保護の利用規約を調査・作成する技術者を主な対象とする。

本資料の構成

1：API 標準設計書の概要

1.1 API 普及に関する課題と API 標準設計活用のメリット

日本での API 導入状況と課題とメリットを示す。

1.2OAS 概説

OAS が構築された経緯と各構成に関する定義方法のガイドを示す。

1.3 OAS で使用する主なツールと特徴

OAS の代表的なオープンツールの特徴と概要を説明する。

1.4API 標準設計書の作成事例

実在する API 標準設計書作成のための要件定義、設計、テスト、リリースまでの作業概要を示す。

2：API 標準設計に関する補足

本資料「1：API 標準設計書の概要」で説明しきれなかった API 標準設計に関する、補足情報を説明する。

3：付録（政府標準を参考にしたサイト利用規約のひな型）

3.1 政府標準を参考にしたサイト利用規約のひな型を活用するメリット

自組織のサイトの利用規約を構築する場合にひな型を活用する場合のメリットを示す。

3.2 API 利用規約

API、データを公開する場合の API 利用規約の目的と概要を示す。

3.3 データ利用規約

API、データを公開する場合のデータ利用規約の目的と概要を示す。

3.4 個人情報保護方針

API、データを公開する場合の個人情報保護方針の目的と概要を示す。

3.5 サイト利用規約のひな型のリンク

サイト利用規約のひな型のリンクを示す。

4：用語集

本資料で使用した用語の説明とリンクを示す。本文中に【番号】で記述する。

1：API 標準設計書の概要

1.1 API 普及に関する課題と API 標準設計活用のメリット

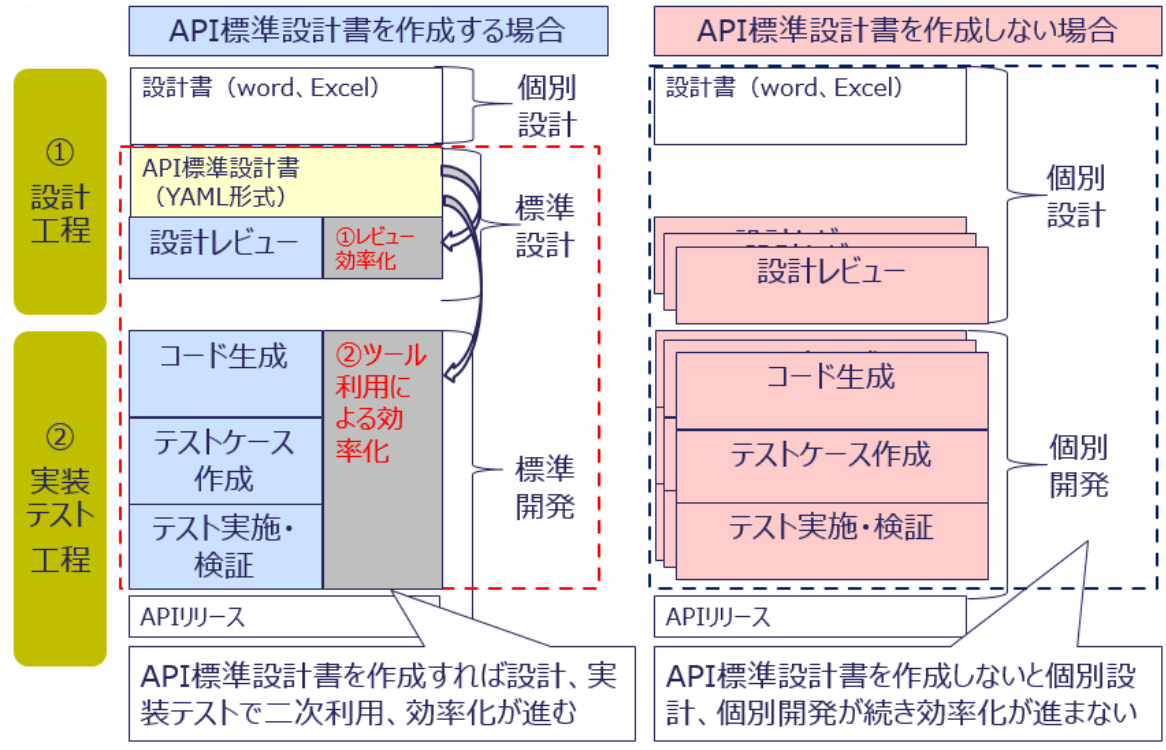
日本では、欧米と比較して API の活用が限定的であり、官民の組織間のデータ連携が十分に進んでいないと考えられる。以下、原因を考察する。

1 つ目の原因は、API 提供者が標準に準拠した設計書を作成していないためと考えられる。API の設計書が標準化されていない場合、フォーマットや記述粒度が統一されず、開発者は設計書の理解や解析に手間がかかる。

2 つ目の原因は、API 標準設計書の読み方及び二次利用方法が API 利用者に十分に理解されていないためと考えられる。API の二次利用が進まないと、開発者は個別設計と個別開発の繰り返しにより手間がかかる。

次に、API 標準設計書を作成する場合、しない場合の設計・実装テスト工程の比較を行い、API 標準設計書を作成するメリットを考える。

図 1：API 標準設計書を作成する場合としない場合の設計・実装テスト工程の比較
（一般的なプログラムの設計、実装・テスト工程を基に考察）



API 標準設計書を作成する場合の各工程でのメリット

①設計工程のメリット

- API 標準設計では設計ツールの記述項目、記述レベルが標準化されているので設計ツールの利用により設計者が自ら、抜けや漏れに気づきやすくなり、設計品質が向上しレビューを効率的に実施できる。
- API の設計が標準化され使いやすくなることで、公開された API を自組織の環境で稼働する場合等の二次利用が促進される。

②実装・テスト工程のメリット

- API 標準設計で広く利用される YAML【5】は人間可読性があり、かつ機械可読性がある設計書である。テストツールを利用することで、設計コードやテストケースの自動生成ができるので実装、テスト工程の品質と開発効率が向上する。
- 公開された API を自組織の環境で二次利用したい場合、設計と実装・テスト工程でツールが利用できるので効率的である。

さらに、API 提供者と API 利用者それぞれの立場からメリットを考える。

表 2：API 標準設計書を作成する場合の API 提供者と API 利用者のメリット

対象	設計工程	実装・テスト工程
API 提供者	• 設計品質の向上 • 設計レビューの効率化	• ツール利用によるコード生成、テスト仕様書の生成により効率が向上する
API 利用者	• 設計品質の向上 • 設計レビューの効率化 • 二次利用の促進	• ツール利用による開発効率の向上 • 稼働実績のある API を利用できるので品質が保証されている • 二次利用の促進

上記から、API 提供者が API 標準設計書を作成する手間が増えるが、API 標準設計書を作成することは、API 提供者、API 利用者の双方にとって、品質と開発効率の向上、二次利用による効率化のメリットがある。さらに欧米と同様に API 利用が進めば業界内、組織間のデータ連携が進むと考えられる。

1.2 OAS 概説

OAS（OpenAPI Specification）【6】

OAS はシンプルで拡張性の高い Web API の設計指針である「RESTful API【7】」の事実上の標準であり、ISO/IEC によって国際標準として認定されている。API の設計からテストまでの工程を標準化し、

ツールを活用してコードやドキュメントを自動生成することで、開発・運用・管理を効率的に行える。国際的に広く利用されているので API の国際的な提供及び利用が可能である。日本の組織でも導入が進みつつある。

OAS 構築の経緯

OAS の原型は、2011 年に SmartBear Software 社が開発した「Swagger」というオープンソースの API フレームワークで、2015 年に設立された OAI（OpenAPI Initiative）【8】に移管された。OAI は、API の標準化を推進し、開発者がより効率的に API を設計・管理できる環境を提供する団体である。2017 年に OAS 3.0 を発表し、より柔軟な API 設計が可能になった。2021 年に OAS 3.1 を発表し JSON Schema【9】との互換性が強化された。現在、OAS は国際的に広く利用されている。

1.3 OAS で使用する主なツールと特徴

OAS で使用する主なツールと特徴を示す。オープンツールなので基本的に無料で利用できる。目的に合わせて利用してほしい。

Apicurio Studio【10】

Design-First（デザイン主導開発）【11】の考えに基づく設計ツールである。Apicurio Studio は GUI でエンドポイント【12】追加、フィールド定義等が可能で、エラーや警告をリアルタイムで表示する。開発者と依頼者が設計工程で画面を見ながら仕様の合意ができる。

Swagger Editor【13】

OAS の API 標準設計書を編集するためのエディタである。画面の項目にツールの指示に従って入力することで、標準に沿った項目毎の定義が可能で設定漏れ・誤りを防ぐことができる。また、入力した項目に関するテストをツール上で実行できるため、設計しながら入力項目の妥当性をテストできる。また、Swagger Editor はテストフォームを自動で生成する。テストフォームはテスト用の入力画面やデータのインターフェースのことで、テスト工程で、入力データ、バリデーション、データ送信結果、エラーハンドリングの検証等に使用される。下記に Swagger Editor のテストフォームの用途・メリットを示す。

- ✓ テストフォームにより、API の動作を検証しつつ、開発できる。
- ✓ 正確性:OAS で定義された内容に基づくため、API 仕様書と実際の挙動が一致する。
- ✓ 簡便性:開発者が手動でリクエストを構築する必要がなく、ボタン操作だけでテストできる。
- ✓ デバッグに便利:実際のリクエスト/レスポンスを可視化できるため、エラーの原因を特定しやすい。

Swagger UI【14】

API 標準設計書は API の設計や定義において、人間可読性があり、かつ機械可読性を持つ YAML が広く利用されている。Swagger UI は API ドキュメントとテストフォームを生成するツールである。API 構築完了後に、テストフォームを利用してリクエストを送信し、API の動作を確認できる。

OpenAPI Generator

OpenAPI Generator は OAS 3.x に対応し、Java, Python, JavaScript, C#, Go, PHP, Ruby 等多数の言語向けた SDK を自動生成するツールである。

1.4 API 標準設計書の作成事例

目的

ここまで API の定義方法について説明してきたが、実践的な視点から捉えるために、実在する API を事例に挙げ、API 標準の設計からテストまでの作業工程を説明する。

前提

- 検索機能を最初に手掛けると設計からテストまでの作業工程が理解しやすいので、国税庁が提供する「法人番号システム WEB-API」【15】を事例として取り上げる。
- 「法人番号システム WEB-API」は法人番号、取得期間、法人名による 3 種類のデータ検索用 API がある。
- 仕様書は WORD 形式の PDF で公開され、API 標準設計書の形式では提供されていない。
- 法人番号システムの WEB-API を利用するためには同サイトでアプリケーション ID の申請が別途必要である。

API 標準設計書作成の作業工程を示す。

1.4.1 設計準備

標準に準拠する設計と項目定義の方針を定義する。

設計方針

- ここでは可読性に優れる YAML を利用する。
- 認証方式、セキュリティスキームは API キー、OpenID Connect【16】等を利用する。
- エラー表現は RFC7807【17】形式を使用し、BadRequestError や NotFoundError を定義する。

項目定義方針

- データフォーマット、非標準表記（;区切り等）を標準化する対象を選定する。
- 項目の定義には、API テクニカルガイドブックで推奨する標準型（ISO8601 日付、UTF-8 文字コード、JIS X 0401/0402 所在地コード等）を採用し、OAS スキーマで型定義を行う。
- 共通データモデルを再利用可能な形にする。これにより重複定義を避け、保守性を向上させる。項目定義の例として日付・日時、地理情報、列挙値の定義方法を示す。
  - ✓ **日付・日時**： format:date-time を使用し、YYYY-MM-DDTHH:mm:ss+09:00 形式とする。
  - ✓ **地理情報**：必要に応じて prefecture_code,city_code 等は type:string + description で JIS コード準拠を記述する。

✓ **列挙値（enum）**：性別コード、言語コード等、標準化されたコードリストが存在する場合、「enum」で特定の値のみ許可する等の制御を行う。これにより OpenAPI Generator や Stoplight がパラメータ値を補完し、バリデーションチェックを行う。

1.4.2 設計

API 標準設計書（YAML）を作成する。Swagger Editor のセクションの項目毎に定義を入力する。

- Swagger Editor を起動する。ここではオンライン版を使用する。ブラウザで Swagger Editor（https://editor.swagger.io/）にアクセスする。
- 定義詳細については、本資料「2：API 標準設計に関する補足」及び「API テクニカルガイドブック【18】」を参考にしていただきたい。
- 本資料と同じフォルダに、国税庁の「法人番号システム WEB-API」の YAML のサンプル設計書を添付したので参考にしていただきたい。組織の目的、環境に合わせて変更した上で利用する必要がある。

表 3：API 標準設計書のドキュメント構成と記述項目、記述概要

セクション	定義の概要	API 標準設計に関する補足
Info	API の設計・運用を体系的に管理するため API のタイトル、バージョン、説明等を定義する。	2.1Info の定義
Servers	API のサーバに関する情報を定義する。	2.2Servers の定義
Paths	API のエンドポイントと対応する操作（HTTP メソッド）を定義する。	2.3Paths と Operations の定義
components	スキーマやレスポンス、パラメータ等の再利用可能な要素を定義する。	2.4Components の定義
Security	API の認証及び認可に関する仕様を定義する。	2.5 セキュリティと認証方式の定義
Tags	API を分類するためのタグを定義する。	2.6Tags, ExternalDocs の定義
externalDocs	外部ドキュメントへのリンクを定義する。	同上

1.4.3 テスト

上記で作成した API 標準設計書の構文が標準に準拠しているかバリデーションチェックを行う。

- バリデーションチェックとは、ツールのバリデーション（入力データや定義情報が適切な形式やルールに従っているか検証する）機能を利用し、作成された API 標準設計書の構文の妥当性をテストすることである。

- バリデーションチェックには Lint（リンター）【19】と呼ばれるバリデーションチェックツールを使用する。代表的な Lint ツールは Swagger Editor や Stoplight Studio【20】である。Lint ツールを複数利用しエラーを検出することで、より正確な検証が可能となる。

Swagger Editor によるテストを行う。

- ここでは Swagger Editor のバリデーションチェック機能を利用する。エラー箇所はハイライト表示される。
- ツールが検出したエラー箇所のエラーメッセージを確認し設計書を修正する。

1.4.4 接続テスト

バリデーションチェックで問題無いことを確認した API 標準設計書を Swagger Editor で API 提供者のサーバと接続テストを行う。

図 2：Swagger Editor で YAML を読み込んだイメージ

Swagger Editor で接続テストを行う。

- Swagger Editor にアプリケーション ID（※）を定義する。
  （※アプリケーション ID は別途申請が必要）
- 入力:必要なパラメータやリクエストボディをフォームに入力する。
- 送信:「Try it out」ボタンをクリックしてフォームをアクティブにする。
- 「Execute」ボタンを押すとリクエストが実行される。
- 検索結果として想定したステータスが返るか確認する。
- 下記以外のレスポンスが返った場合は「2.7 エラー対応」でエラー内容を解析し対応する

ステータス	内容	説明
200	OK	リクエスト成功

1.4.5 公開用 SDK 生成

テスト終了後、openapi-generator 等を使用し、各組織の環境に合わせた SDK を生成し API 利用者に提供する。

- SDK(Software Development Kit)は、API 開発者が自組織の利用者向けに提供する API を利用するためのツール群である。ライブラリ、コードサンプル、ドキュメント、ユーティリティ等を含む。

2：API 標準設計に関する補足

2.1 Info の定義

info の定義：API 利用者が API の概要を理解できるよう、info の各項目を下記の要領で記述する。

- title は API のタイトルを記述する。
- version は"`1.0.0`"等セマンティックバージョニング【21】を使い、現在のバージョンを記述する。
- `description` は API の目的、関連法令、データ更新頻度、使用上の注意等を記述する。
- `contact` には問い合わせ窓口メール・URL を記述する。
- license には利用規約のリンク等を記述する。

description に政府相互運用性フレームワーク（GIF）【22】準拠を記述する。

`description` には、API が政府相互運用性フレームワーク（GIF）の原則に基づき設計されている旨を記述する。参照先の URL を `externalDocs` に添付する。これにより API 利用者は背景を理解しやすくなる。

OAS 定義ファイル公開場所や変更履歴を記述する。

`info.description` または `externalDocs` で公開用 URL、GitHub リポジトリ、CHANGELOG.md へのリンクを提供することで、API 利用者に更新履歴や最新版の取得方法を伝える。

2.2 Servers の定義

サーバセクションで、API が提供されるサーバの URL や説明を試験環境/本番環境毎に記述する。

例：

`servers` に本番と試験環境を併記：

```yaml
servers:
  - url: https://api.example.com/v1
    description: 本番環境
  - url: https://sandbox.api.example.com/v1
    description: テスト・検証用環境
```

URI の設計ルール

URI 設計ルールは API の「使いやすさ」と「管理しやすさ」に配慮する。

- 直感的で理解しやすい API を提供：開発者が URI を見ただけで、どのリソースにアクセスできるのか理解できる。
- 一貫性のある設計を維持：プロジェクト全体で統一した設計ルールを適用し、API の一貫性を維持する。
- 拡張性と保守性を向上：URI の設計を統一することで、API の保守やバージョン管理がしやすくなる。
- 可読性を高める：シンプルで直感的な URI にすることで、API 利用者が理解しやすくハイフン（-）を使い、単語の区切りを明確にする。

2.3 Paths と Operations の定義

Paths は、API エンドポイント（クライアントとサーバの通信を行う URL）の一部を定義し、リソース（API が操作する対象のデータ単位）を指定する。

Paths は以下の用途・メリットがある。

- リソースの識別：API エンドポイントを通じてどのリソースにアクセスするかを定義する。
- リソースの階層構造の表現：リソース間の階層構造を URL パスで表現する。
- 可読性の向上：明確なパスを使用することで、API 利用者の可読性が向上する。

Operations は、特定の HTTP メソッド（GET、POST、PUT、PATCH、DELETE 等）を使用してリソースに対する具体的な操作を定義する。HTTP メソッドを意味通り正しく割り当てることで標準に準拠するので API 利用者が使いやすくなる。

表 4：HTTP メソッド定義

HTTP メソッド	定義
GET	リソース取得（一覧・詳細）
POST	新規作成または非同期タスク開始
PUT	全置換更新（リソース全項目更新）
PATCH	部分更新
DELETE	リソース削除

2.4 Components の定義

Components は API の設計を効率化し、統一性を持たせるために再利用可能な要素を一元管理するためのセクションである。

表 5：Components で使用する項目一覧

項目	定義	用途・メリット
Schema	API で使用するデータモデル、エンティティの構造を定義し、リクエストやレスポンスで利用されるデータ形式を統一する。	リクエストボディやレスポンスのフォーマットを統一し、API のデータ構造を標準化する。
Response	API の標準的な応答（成功・エラー・認証エラー・リソース未検出等）、レスポンスのフォーマットを統一する。	統一されたレスポンスフォーマットを提供し、開発者がエラーハンドリングしやすくする。
Parameter	クエリパラメータ、パスパラメータ、ヘッダパラメータ、クッキーパラメータ等を定義する。API のリクエスト時に外部から渡される変数の型や制約を指定する。	API のリクエストに必要なパラメータの形式や制約を明確にし、リクエストの仕様を統一することで再利用しやすくなる。
Example	API リクエスト・レスポンスのサンプルを定義する。開発者が、具体的なリクエスト・レスポンスの形式、参照データを提供する。	API 仕様書において、リクエストやレスポンスの具体例を提供し、API 開発者が仕様を理解しやすくする。
Request Body	POST や PUT 等のメソッドで送信するリクエストボディのデータ構造を定義する。API のリソース作成や更新の際に、必要なデータのフォーマットを明示する。	リクエストボディのデータ形式を統一することで API のリソース作成・更新時に必要な情報を明確化する。

Header	HTTP ヘッダを定義する。認証情報、コンテンツタイプ、キャッシュ制御等、API リクエスト・レスポンスの共通メタデータを指定する。	API リクエスト・レスポンス時に必要なヘッダ情報を標準化することで通信を適切に制御する。

2.5 セキュリティと認証方式の定義

セキュリティと認証は API の利用者を適切に認証し、不正なアクセスを防ぐための仕組みを設計・実装する。

セキュリティ定義

- TLS/HTTPS の使用:すべての API 通信に TLS（Transport Layer Security）を使用することで、データの盗聴や改ざんを防ぐ。API エンドポイントは HTTPS で提供する。
- HSTS（HTTP Strict Transport Security）:HSTS ヘッダを使用することで、ブラウザに対して常に HTTPS を使用するよう指示する。これにより、ダウングレード攻撃を防ぐ。
- レートリミットの設定:API への過剰なリクエストを防ぐため、一定時間内に許可されるリクエスト数を制限する。
  ✓ 例: 1 分間に 5000 リクエストまで許可する。
- CORS（Cross-Origin Resource Sharing）の設定:ブラウザからのクロスオリジンリクエストを制御し、特定のオリジンからのアクセスのみを許可する。
  ✓ 例: Access-Control-Allow-Origin: https://trusted-origin.com

認証方式

- API キーの使用：クライアントが API にアクセスするために必要な一意のキーで、API キーはヘッダ、クエリパラメータ、またはボディに含めて送信される。
- OAuth 2.0 の使用：権限の委譲を実現するための業界標準プロトコルで、クライアントに対してリソースオーナーのリソースへのアクセスを許可するトークンを発行する。
- OpenID Connect の使用：OAuth 2.0 の上に構築された認証レイヤーで、ユーザのアイデンティティを検証する。openid スコープを使用して、ユーザ情報を取得する。

デジタル庁の API テクニカルガイドブックでは、認証方式として OpenID Connect の利用が推奨されている。

2.6 Tags, ExternalDocs の定義

タグの定義方法

tags セクションでタグを宣言し、各エンドポイントの tags フィールドで適用する。

タグの用途・メリット

- API エンドポイントの分類：タグを設定することで、API を機能ごとに整理し、関連するエンドポイントをグループ化できる。これにより、API の構造が明確になり、管理効率が向上する。

- API ドキュメントの可読性向上：Swagger UI 等の API ドキュメントツールにおいて、タグを使用することでエンドポイントが整理され、検索がしやすくなる。また API の検索・フィルタリングができる。

ExternalDocs の用途・メリット

✓ ExternalDocs（外部ドキュメント）は、API に関する追加情報や詳細な説明を提供する外部リソース（GitHub リポジトリ、公式ドキュメント、サポートページ等）へのリンクを提供する。

例：

```
externalDocs:
  description: Find more info here
  url: https://example.com/docs
```

2.7 エラー対応

エラー対応は API 利用者がエラーの原因を理解しやすい形式で記述し、広く利用されている RFC7807 に準拠する。

例：

```
{
  "type": "https://example.com/probs/invalid-param",
  "title": "Invalid parameter",
  "detail": "The 'offset' parameter must be a positive integer.",
  "instance": "/path/to/endpoint"
}
```

表 6：HTTP ステータス標準

ステータス	内容	説明
200	OK	リクエスト成功
400	Bad Request	リクエストデータに問題あり（パラメータの内容や文字コードが誤っている等）
401	Unauthorized	認証されていない
403	Forbidden	アクセスを拒否された（利用制限超過等）
404	Not Found	URL にリソースが存在しない
429	Too Many Requests	サーバ側での処理中にエラーが発生

Lint エラーへの対応

- Type が不一致（string なのに例が数値）は Swagger Editor で検知する。
- 未定義項目に対する参照（$ref）は Stoplight Studio で検知する。
- 不正な値や項目抜けは Apicurio や Stoplight の Lint で検知する。

デプロイ時のエラー対応

- **CORS エラー：** CORS エラーは API 利用者が自身のオリジン（スキーム、ホスト、ポートの組み合わせ）とは異なるオリジンにリクエストを送信し、サーバ側がそのリクエストを許可しない場合に発生す

る。ツール等で CORS エラーの詳細を確認する。サーバで「Access-Control-Allow-Origin」を定義する。サーバで対応しない場合は利用者側の環境をサーバ側の環境に合わせる。

- **認証エラー**：OpenID Connect 設定 URL 誤記やトークン不備をログで診断、設定ファイル・OAS の openIdConnectUrl を確認する。

3：付録（政府標準を参考にしたサイト利用規約のひな型）

3.1 政府標準を参考にしたサイト利用規約のひな型を活用するメリット

自組織のサイトで API、データを提供する場合にはサイト上に API 利用規約、データ利用規約、個人情報保護方針等を掲載する必要がある。

表 7：サイトで定義する必要のある利用規約の種類と目的

利用規約の種類	利用規約の目的
API 利用規約	API を適正に利用し、不正利用防止、責任範囲の明確化、API、データ等の管理ルールを定義する。
データ利用規約	データの適正な利用促進のため、商用利用範囲の明確化、不正利用禁止等のルールを定義する。
個人情報保護方針	法令順守により、個人情報の管理、API 提供者等の権利保護、データ漏えい防止等のルールを定義する。

一般に、サイト利用規約を定義するには関連法規の調査、法律の専門家への問い合わせ等の手間がかかる。利用規約のひな型を活用すると、関連法規の調査や法律の専門家への問い合わせを減らすことができる。また、自組織に合わせたカスタマイズが可能で、効率的かつ柔軟に自組織に合わせた利用規約を構築できる。

表 8：利用規約を活用する場合の利点と活用しない場合の課題

項目	活用する場合の利点	活用しない場合の課題
統一ルールによる効率化	利用規約により、組織内でルールが統一されるため、利用者が迷わずに API、データの利用条件を理解できるので効率的である。	利用規約や、標準が無いと組織内で異なるルールが存在するため利用者が個別に条件を確認することで利用者からの問い合わせ対応等の負荷が発生する。
リスク回避	運用実績のある利用規約により利用者による著作権侵害等の法的リスクを回避できる。	利用者が意図しない著作権侵害等の法的リスクが高まる。
二次利用	国際的に広く利用されているクリエイティブ・コモンズ・ライセンス【23】に準拠した表記をすることで、API、データの二次利用及び国際的な相互運用が促進される。	各組織で異なる二次利用に関するルールがあると、API,データの組み合わせ利用等が難しくなる。二次利用、国際的な相互運用は促進されない。

上記より、利用規約の活用により、API、データ利用の効率化、リスク回避、二次利用の促進、国際相互運用性が促進されると考えられる。

3.2 API 利用規約

API 利用規約の目的

API 利用規約の目的は、API の提供に関して利用者が安心して利用できる環境を確保し、官民の組織が提供する API の信頼性を高めることである。また、利用者側も明確なルールの基で API を活用できるため、法的なリスクを回避することができる。

API 利用規約の概要

表 9：API 利用規約の主要項目と記載事項（概要）

主要項目	記載事項（概要）
適用範囲	利用の範囲（商用利用可否、二次利用の可否等）
利用目的	API の利用目的、利用条件、API 利用者と提供者の関係
利用手続き	API の利用手続き
認証とアクセス管理方式	API キーや OAuth 等の認証方式、アクセス制限
データの取り扱い	取得したデータの保存・加工・共有の可否、個人情報の取り扱いルール
個人情報の取り扱い	API 提供者等の個人情報取り扱いルール
利用制限	API のリクエスト回数制限、禁止行為（仕様としての禁止事項、過剰アクセス制限、リバースエンジニアリング禁止等）
責任範囲と免責事項	API 提供者の責任範囲、障害発生時の対応、損害賠償責任有無
API 利用規約改訂時の対応	API 利用規約の変更があった場合の通知方法と適用タイミング
準拠法と紛争解決	準拠法（日本法等）、紛争時の管轄裁判所の指定

3.3 データ利用規約

データ利用規約の目的

データ利用規約の目的は、データ提供者や著作権者が著作権を有し、二次利用可能な著作物のデータ利用を促進すること、ならびにクリエイティブ・コモンズ・ライセンスと互換性を持たせ、国際的なデータの利活用を促進することである。また、データ利用規約の適用範囲や利用条件を明確にし、利用者の混乱を防ぎ、第三者の権利を侵害する場合や、個別法令による制約が生じる場合の対応を明確にすることである。

データ利用規約の概要

表 10：データ利用規約の主要項目と記載事項（概要）

主要項目	記載事項（概要）
コンテンツの利用	当ウェブサイトで公開しているデータ等の情報（以下コンテンツと表現する）は、原則、自由に利用可能で商用利用も可とする。
出典の記載	コンテンツ利用時は出典を記載する。編集・加工時はその旨を記載し、未加工データとして誤認させない。
第三者の権利	一部のコンテンツには第三者の権利が含まれる可能性があり、利用者の責任で許諾を得ることが必要である。
個別法令の制約	個別法令による利用制限がある場合は利用者の責任で許諾を得ることが必要である。
適用外のコンテンツ	シンボルマーク、ロゴ、キャラクターデザインや、個別に利用ルールが記載されているコンテンツはデータ利用規約の対象外とする。
準拠法と管轄	本データ利用規約は日本の法律に基づく。紛争はコンテンツを公開している組織の所在地を管轄する裁判所とする。
免責事項	データ等のコンテンツを所有する組織は、コンテンツ利用による行為について責任を負わない。
その他	本データ利用規約は引用等の著作権法上の利用を制限しない。また、クリエイティブ・コモンズ・ライセンスと互換性がある。

3.4 個人情報保護方針

個人情報保護方針の目的

個人情報保護方針の目的は官民のサイトでサービス提供に必要な範囲で個人情報を収集し、その利用目的の範囲内で適切に管理・運用するルール、及び、個人情報の漏えい、滅失、改ざん等を防ぐための必要な措置を講じ、法令に基づく開示要請等の特別な場合を除き、収集した情報の目的外利用や第三者提供を禁止する等のルールを定義することである。

個人情報保護方針の概要

表 11：個人情報保護方針の主要項目と記載事項（概要）

主要項目	記載事項（概要）
適用範囲	対象範囲と対象外の範囲の記載
基本的考え方	個人情報を取り扱う目的と利用者の情報を収集する範囲、取り扱い方法の記載
収集する情報の範囲及び利用	収集する個人情報（利用者情報等）と利用目的（利用者フォル

目的	ダの作成・管理等）の記載
利用及び提供の制限	目的外利用・第三者提供の場合の対応、個人情報の処理を第三者に委託する場合の対応
安全確保の措置	漏えい・滅失・改ざんを防ぐための必要な措置
自己情報の開示	開示請求がある場合の対応

3.5 サイト利用規約のひな型のリンク

表 12：政府標準を参考にした API、データ、個人情報保護方針に関するサイト利用規約のひな型のリンク

タイトル	提供	発行日	内容	リンク
API 導入実践ガイドブック付録利用規約テンプレート及び解説	デジタル庁	2022 年 3 月 31 日（付録の追加日）	デジタル庁の政府相互運用性フレームワーク（GIF）で定義された API 利用規約のひな型と解説である。	24
「公共データ利用規約（第 1.0 版）」の解説	デジタル庁	2024 年 7 月 5 日	公共データに関する利用規約である。地方公共団体等の公的機関でも適用可能である。	25
参考：データ連携基盤規約 Ver.1.0	経済産業省	2024 年 4 月	「データ連携基盤規約」のひな型である。	26
参考：データ連携のためのモデル規約　解説と論点整理	経済産業省	2024 年 6 月	「データ連携のためのモデル規約」の解説と論点整理である。	27
個人情報保護方針のひな型	-	-	個人情報保護方針に関する利用規約のひな型は未提供（今後の対応については検討中）	-

4：用語集

1第 2 次決済サービス指令（PSD2: Payment Services Directive 2）：EU 域内の決済サービスを統一し、競争を促進しながらセキュリティを強化することを目的とした指令である。加盟国ごとの国内法として定義される。

2 **BIAN（Banking Industry Architecture Network）**：金融業界の相互運用性を促進するための共通のアーキテクチャフレームワークを確立、推進することを目的とした、独立した非営利の団体である。オープンソースのツールを提供し、業界全体での標準化を支援する。

3 オープン API をめぐるわが国の現状と展望（デジタル庁 2023 年）
https://www.fisc.or.jp/document/fintech/file/FinTech_20230228_05.pdf?utm_source=chatgpt.com

4 DX 白書 2023（IPA）：DX 白書 2021 と 2023 年版（IPA）である。日本のデジタルトランスフォーメーション（DX）の現状と課題を分析し、企業の DX 推進に向けた戦略を提案している。
https://www.ipa.go.jp/publish/wp-dx/dx-2023.html

5 **YAML**：YAML は人間可読性がある（人間が読みやすい構文を持つ）、かつ機械可読性を持ち、設定ファイルやデータ交換フォーマット、API 設計書として国際的に利用されている。API の標準構文をツールがサポートしているので設計で抜け、漏れが生じにくい。機械可読性があるのでコードの自動生成やツールを利用したテストが行える。

6 OAS(OpenAPI Specification)：RESTful API の設計や仕様を記述するための標準フォーマットである。API のエンドポイント、データ構造、リクエスト/レスポンス形式、認証方法などを、機械可読かつ人間にとって理解しやすい形式である。

7 **RESTful API**：REST（Representational State Transfer）は、クライアントとサーバがリソースの「表現」としてのデータを「転送」しながら、システムの「状態」を変えていく仕組みであり、Web API の設計指針である。RESTful API は、シンプルかつスケーラブルな Web サービスを構築するための設計原則であり、次の 6 つの原則に基づく。①クライアントとサーバを分離し、それぞれ独立して開発できるようにする②各リクエストを独立させ、サーバがリクエスト間の状態を保持しない（ステートレス）③クライアントや中間サーバで API レスポンスをキャッシュ可能にし、パフォーマンスを最適化する④シンプルで一貫性のある設計ルールを維持し、理解しやすい API とする⑤API の内部構造を分離し、スケーラブルなシステムを構築する⑥クライアントに実行可能なコードを提供する

8 OAI：OAI（OpenAPI Initiative）：API の標準仕様である OAS を管理・推進する団体である。2015 年に設立された。https://www.openapis.org/

9 JSON Schema：JSON は JavaScript（プログラミング言語）のオブジェクトの書き方を参考に作られたデータフォーマット(データの記述形式)である。JSON Schema は JSON データの構造やバリデーションルールを定義するための言語で、JSON データの品質管理やドキュメント化、データの整合性確保などの目的で使用される。

10 Apicurio Studio：ブラウザベースのグラフィカルエディタであり、OAS 3.1 に対応している。直感的な UI により、設計段階から OAS に準拠した定義が可能である。https://www.apicur.io/

11 Design-First：ソフトウェア開発や API 開発において、コードを書く前にまずシステムや API の設計を行い、その設計に基づいて開発を進める開発手法である。

12 **エンドポイント**：API が提供する機能やデータにアクセスするための窓口となる URL である。

13 Swagger Editor：API の開発プロセス全体をサポートするツールであり、特に API の設計やドキュメント作成に使用される。https://editor.swagger.io/

14 Swagger UI： API 標準設計書を基に、視覚的な API ドキュメントとテストフォームを生成するツールである。https://swagger.io/tools/swagger-ui/

15 **「法人番号システム Web-API」**：国税庁が管理する法人情報を検索できるシステムである。https://www.houjin-bangou.nta.go.jp/webapi/

16 OpenID Connect：OpenID Connect（OIDC）は、OAuth 2.0 を基盤とした認証プロトコルであり、ユーザ認証（Authentication）を安全かつ標準的な方法で行う。

17 RFC7807：HTTP API のエラーレスポンスのフォーマットを標準化するための仕様であり、IETF（Internet Engineering Task Force）が定めた技術仕様である。https://tools.ietf.org/html/rfc7807

18 API テクニカルガイドブック： API の開発に当たり技術的に考慮すべき事項や留意点を記したガイドブックである（デジタル庁）。https://github.com/JDA-DM/GIF/commit/e5b98cbf6de3dffda5fb4b8c3343641c28661991

19 Lint ツール：Swagger Editor や Stoplight Studio など OAS 仕様の設計データを解析するツールで、コードのスタイル、構文エラー、潜在的なバグ、非推奨の構文などを検出するツールである。

20 Stoplight Studio：設計志向の API 開発を促進し、チームでのコラボレーション、ドキュメント生成、モックサーバ利用など、API ライフサイクル全体を効率化するツールである。https://stoplight.io/studio

21 セマンティックバージョニング：ソフトウェアのバージョン番号の規則を定める広く受け入れられた標準である。メジャー.マイナー.パッチ（例: 1.0.0）のように表記する。

22 **政府相互運用性フレームワーク（GIF）**：政府相互運用性フレームワーク（GIF: Government Interoperability Framework）は、政府機関間でのデータ連携やシステム連携を円滑に行うための基準やガイドラインを定めた枠組みである。（デジタル庁提供）
https://www.ipa.go.jp/digital/architecture/reports/formulation_financial-datamodel.html

23 クリエイティブ・コモンズ・ライセンス：著作権者が自分の著作物の利用ルールを定義し、共有・再利用できるライセンスで世界中で利用されている。非営利団体のクリエイティブ・コモンズ（Creative Commons）により運営されている。「CC BY」はクレジット表記を条件に利用・編集・再配布が可能であり、最も自由度が高い。
https://creativecommons.org/licenses/by/4.0/legalcode.ja

24 API 導入実践ガイドブック付録・利用規約テンプレート及び解説：API 導入実践ガイドブックに付録として追加された API の利用規約テンプレートと解説である。（デジタル庁提供）
https://github.com/JDA-DM/GIF/blob/main/460_%E5%AE%9F%E8%B7%B5%E3%82%AC%E3%82%A4%E3%83%89%E3%83%96%E3%83%83%E3%82%AF/docx/464-1_API%E5%B0%8E%E5%85%A5%E5%AE%9F%E8%B7%B5%E3%82%AC%E3%82%A4%E3%83%89%E3%83%96%E3%83%83%E3%82%AF.docx

25「公共データ利用規約（第 1.0 版）」の解説：政府や地方公共団体が公開するデータの利用条件を明確にし、データの二次利用を促進することを目的に策定された規約である。（デジタル庁提供）
https://www.digital.go.jp/assets/contents/node/basic_page/field_ref_resources/f7fde41d-ffca-4b2a-9b25-94b8a701a037/b44a7e0c/20240705_resources_data_outline_07.pdf?utm_source=chatgpt.com

26 データ連携基盤規約 Ver.1.0：データ連携基盤の利用に関する基本的なルールを明確にし、データ提供者とデータ利用者の間で適切なデータの提供と利用を促進することを目的とする規約である。（経済産業省提供）
https://www.meti.go.jp/policy/mono_info_service/digital_architecture/model_kiyaku.pdf

27 **データ連携のためのモデル規約　解説と論点整理**：「データ連携のためのモデル規約」の解説と論点整理である。（経済産業省提供）
https://www.meti.go.jp/policy/mono_info_service/digital_architecture/moderukiyakukaisetu.pdf

独立行政法人**情報処理推進機構**
Information-technology Promotion Agency, Japan

＜本資料のご利用にあたって＞

本資料の内容を適用した結果生じたこと、また、適用できなかった結果については、IPA は一切の責任を負いかねますのでご了承ください。