国土交通省　地域交通DX推進プロジェクト「COMmmmONS（コモンズ）」
2025年度　シェアサイクルポート共有API標準化プロジェクト
シェアサイクルポート共有API 技術検証レポート付録

シェアサイクルポート共有API
システム設計書

国土交通省　総合政策局　モビリティサービス推進課

No.018-01

目次

1. 開発スコープ

1-1. 概要

シェアモビリティサービスにおける複数事業者間のポート共有を実現するための「ポート共同利用インターフェース」を標準 API として開発し、これを用いた「ポート・モビリティ共有システム」を開発する。

本システムは、複数のシェアモビリティ事業者（以下「事業者」という。）がそれぞれ管理する駐輪ポートとモビリティを、他事業者の利用者が利用可能とするための共通的な業務モデル及びシステム仕様を標準化し、実装するものである。具体的には、ポート情報、モビリティ情報、貸出・返却可能台数といったポートステータスおよびモビリティステータス、貸出予約・返却予約などを API ベースで連携・同期させることを目的とする。

なお、以下の図の「ポート・モビリティ共同化による新規の移動パターン」が今回実現する移動パターンである。

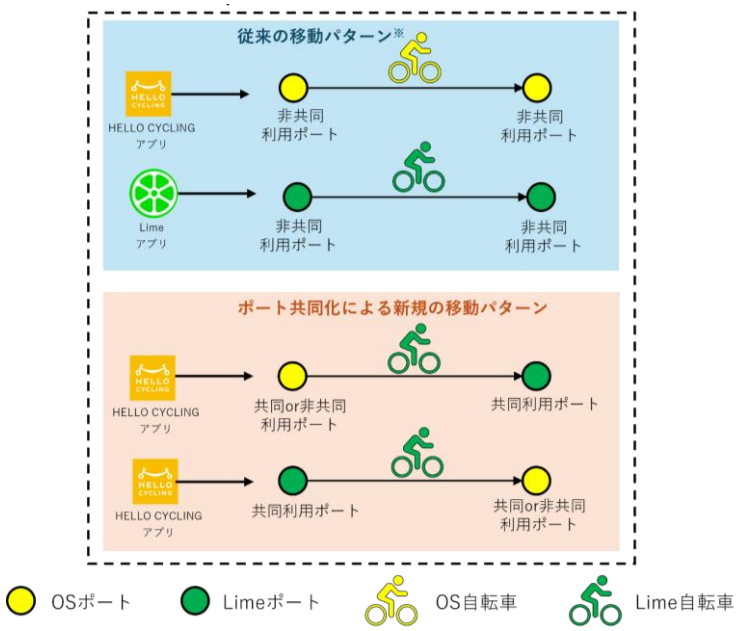


図 1-1 ポート・モビリティ共有により実現する移動パターン

システム実装に先立ち、事業者間のポート・モビリティの共同利用を可能とするための業務モデルとデータ連携仕様を策定する。共同利用の対象となるポートは、物理的には一つの地物でありながら、複数事業者の利用を前提とするため、各事業者のシステムにおけるポート ID のマッピングルールやポートの属性情報（名称、位置、利用時間、設置容量等）を標準化されたスキーマで相互共有する必要がある。

本システムにおいては、国際的なシェアモビリティデータ標準である GBFS（General Bikeshare Feed Specification）に準拠しつつ、ポートの共同運用を目的とした独自拡張項目（利用可能な事業者識別子、予約連携可否、精算区分等）を設ける。

ポート共有システムは、以下の主要機能で構成される。

1. 情報連携インターフェース機能

事業者ごとに管理されているポート情報、モビリティ情報、およびステータス情報（空き状況、予約状況等）を、共通 API および共通メッセージ仕様を通じて取得・提供する機能である。これにより、利用者は利用しているシェアサイクルアプリ上で他事業者ポートの空き情報を確認し、自社・他社問わず、車両の予約及び利用を行うことが可能となる。情報連携は REST API 形式および MQTT 形式で行い、データの提供タイミングには Publish-Subscribe 方式を想定する。

本システムは、クラウドサービスを基本として開発され、クラウド環境での可搬性・拡張性を備える。また、外部 MaaS 事業者との API 連携も可能とする設計とし、今後の社会実装における標準的インフラとして展開されることを想定している。

1-2. システムを利用する業務全体像とシステム利用フロー

1. 業務フロー

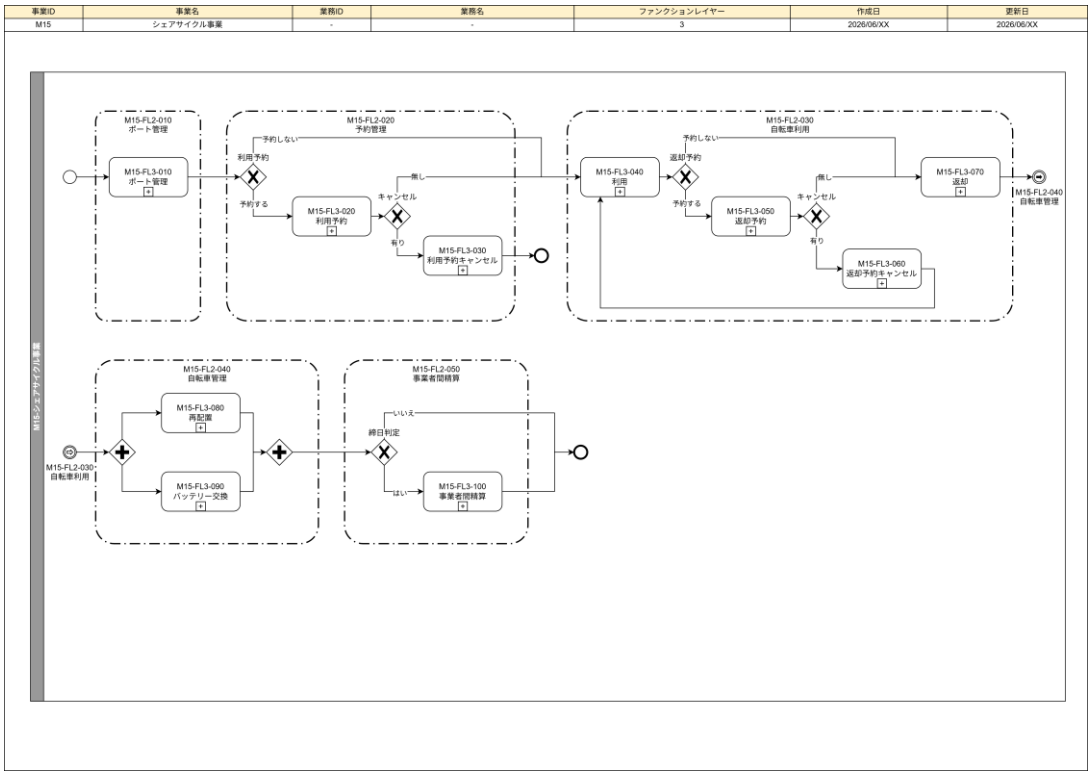


図 1-2 業務フロー全体像

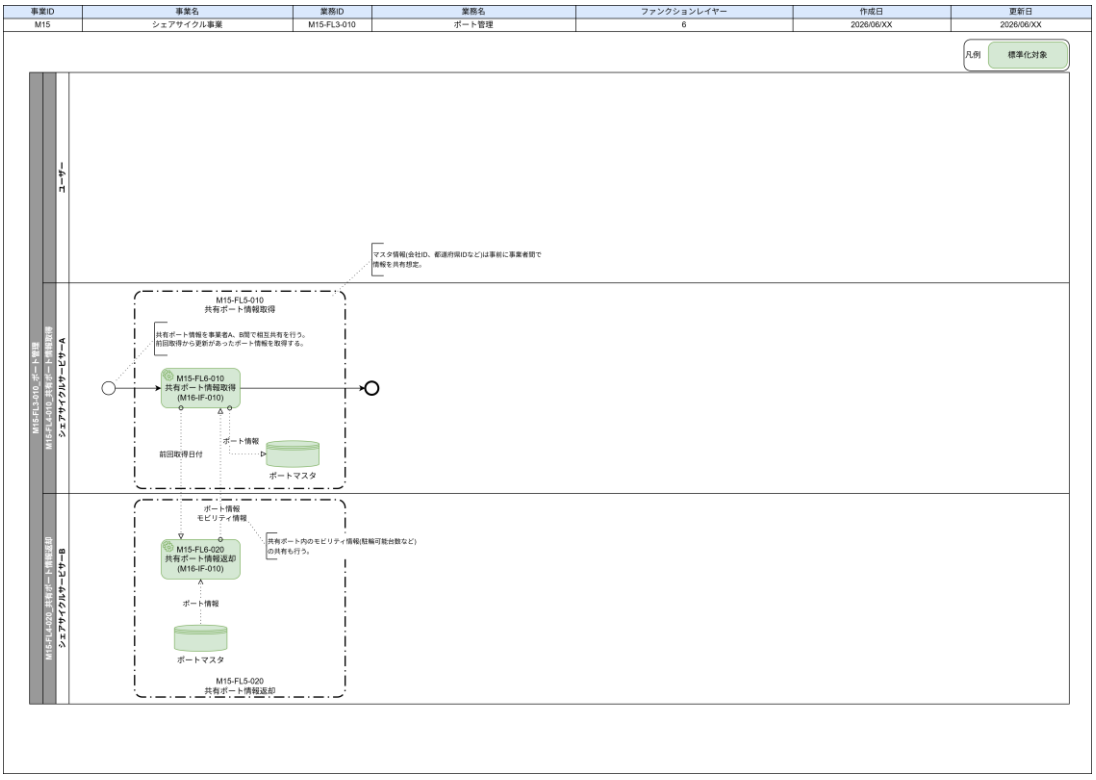
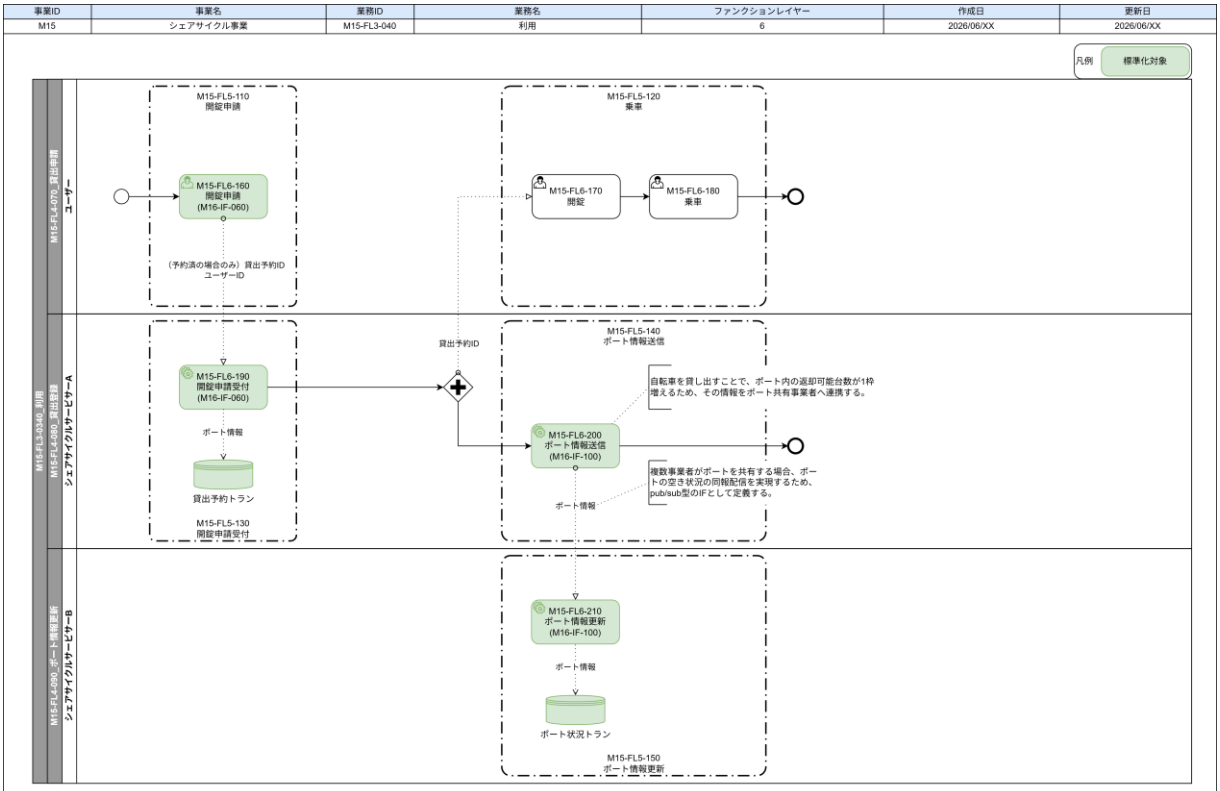


図 1-4 業務フロー詳細（ポート管理）

図 1-5 業務フロー詳細（利用予約）



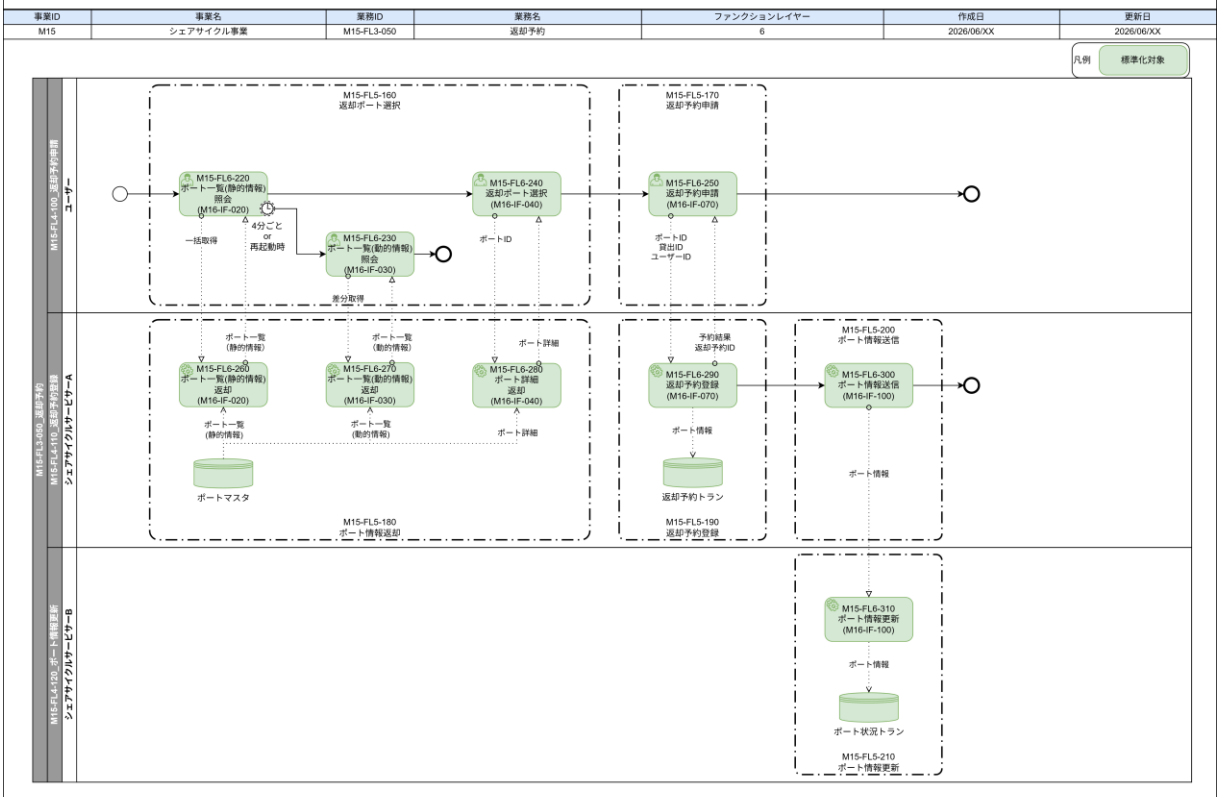


図 1-6 業務フロー詳細（自転車利用）

図 1-7 業務フロー詳細（自転車利用）

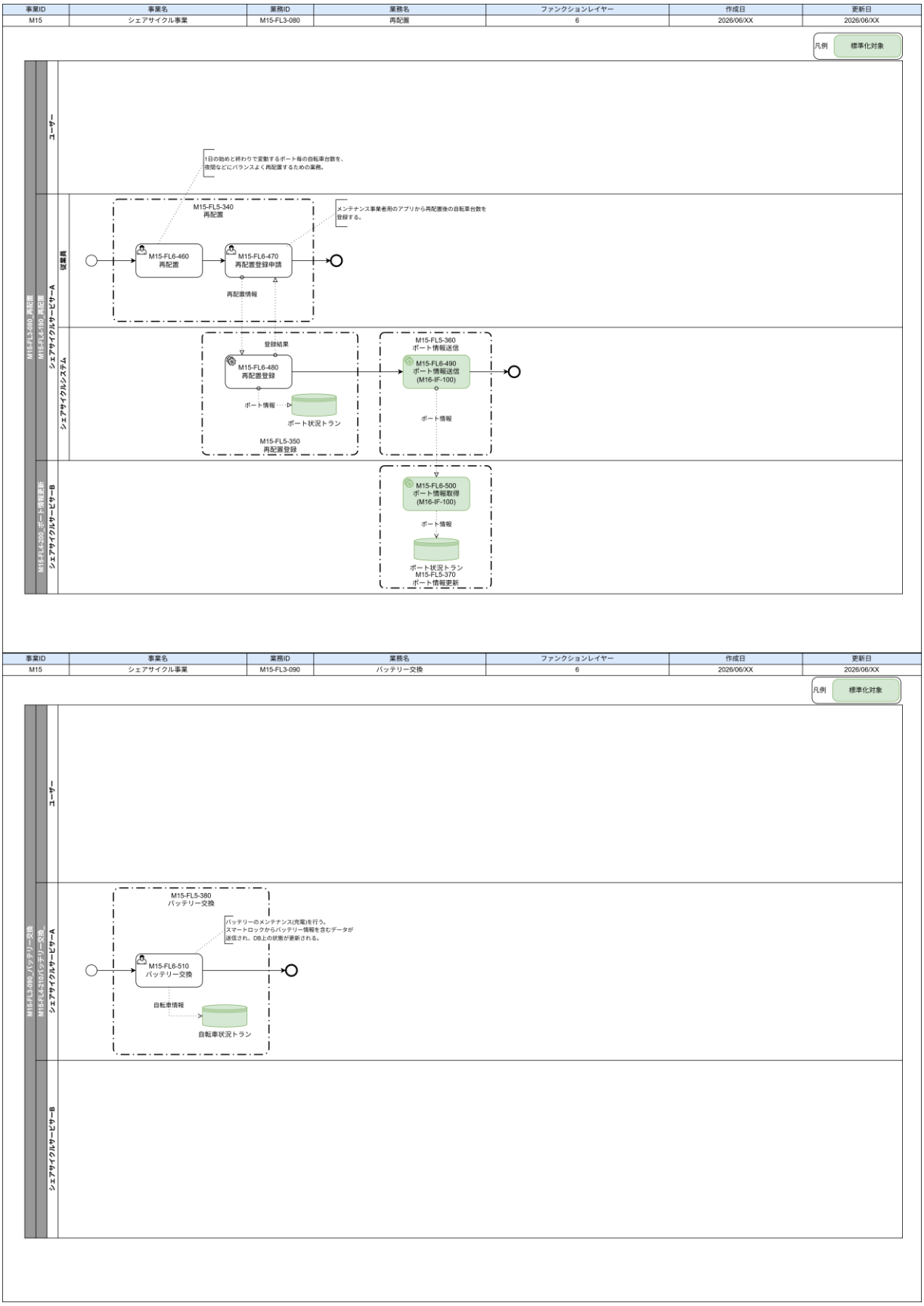


図 1-8 業務フロー詳細（自転車管理）

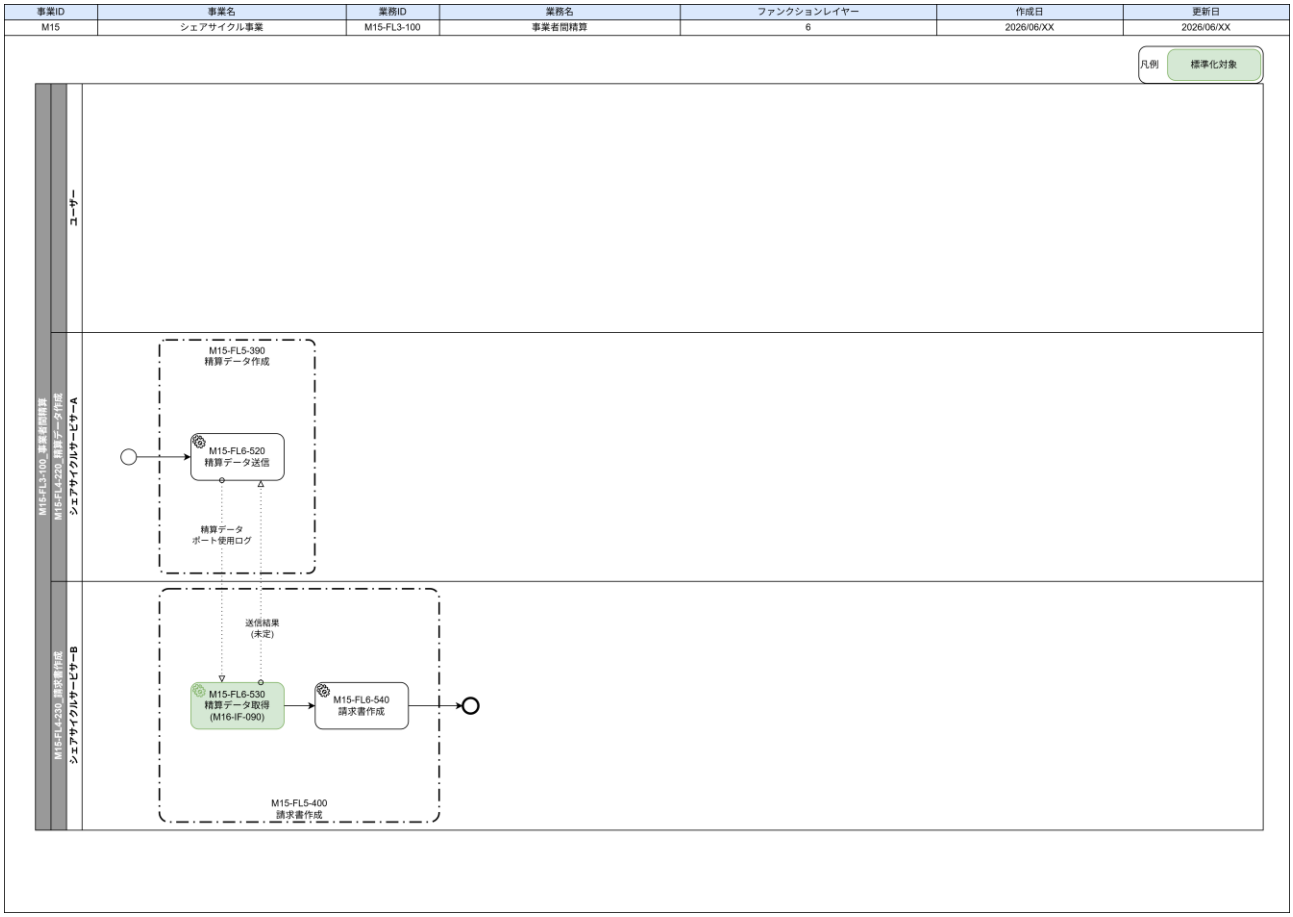


図 1-9 業務フロー詳細（事業者間精算）

2. システムシーケンス図

① ユーザー

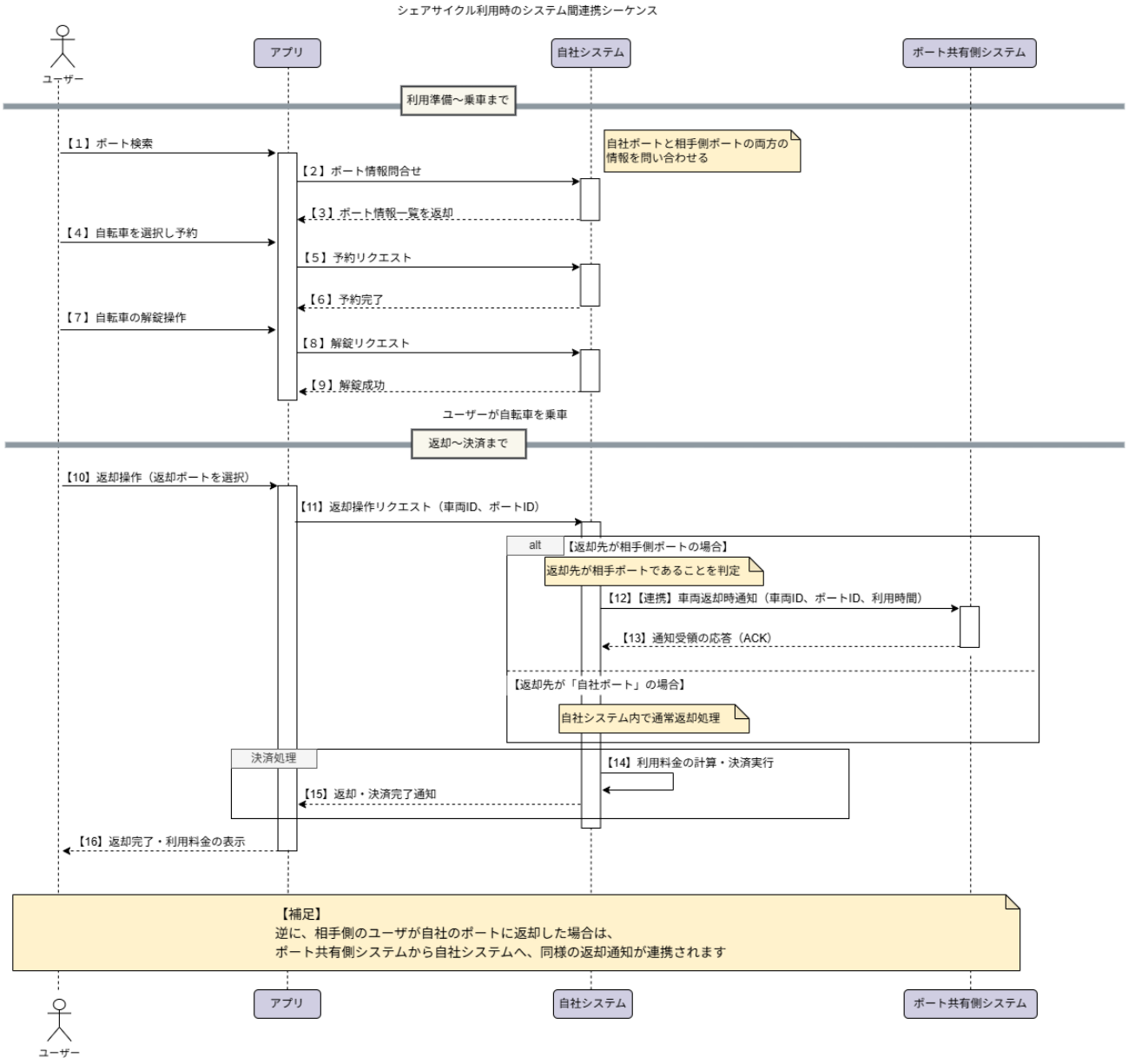


図 1-10 駐輪スペースの選択・予約、誘導処理

② 共同利用ポートの準備

・事業者間データ共有

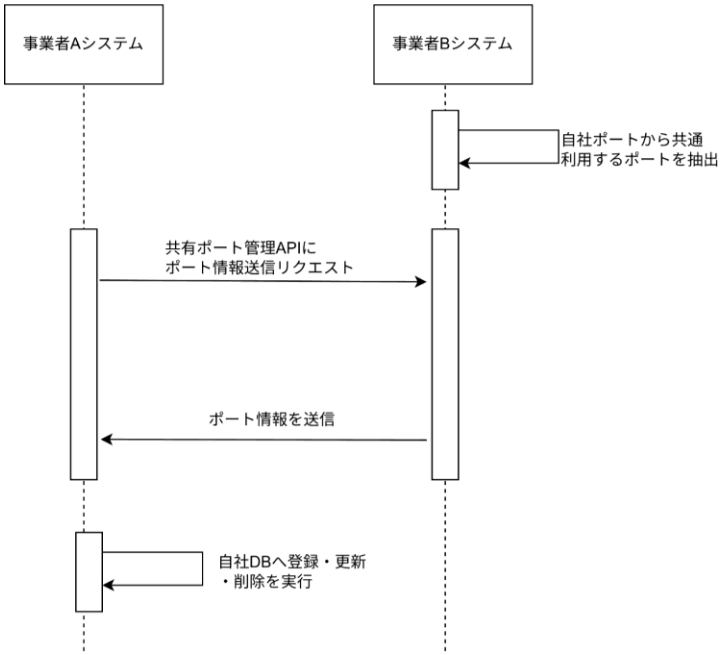


図 1-11 事業者間データ共有

・事業者間精算

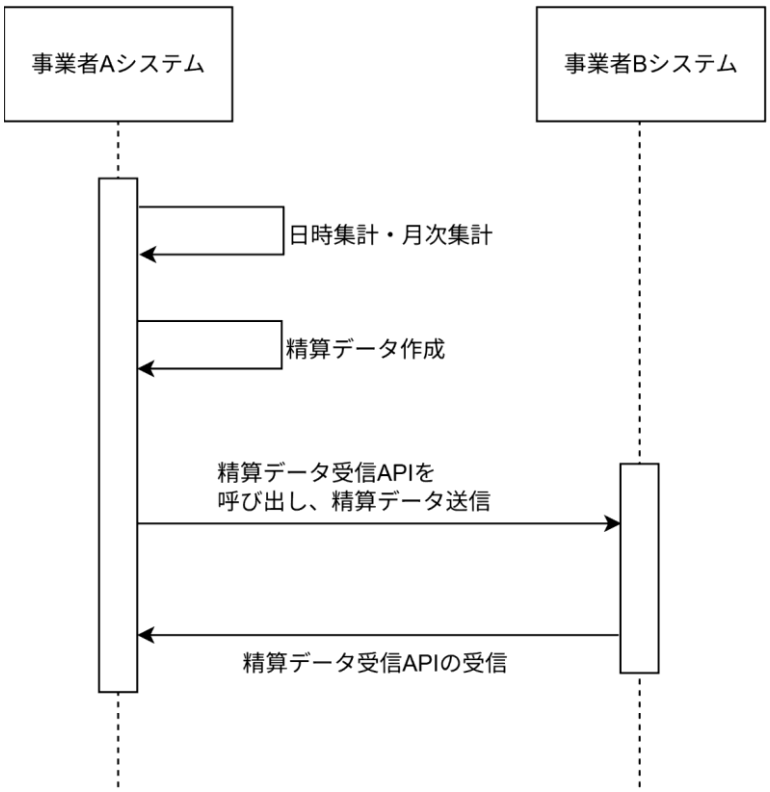


図 1-12 事業者間精算

2. シェアサイクルポート共有 API システム
：機能要件（FN/SL/AL/CO/HW/IF/UI）

2-1. システム機能（FN）

2-1-1. システムアーキテクチャ

3 社以上によるポートの共同利用を見据え、N 対 N の情報連携を柔軟に実現可能なアーキテクチャとして、メッセージブローカを介してポートステータスを共有する Pub/Sub 型アーキテクチャを採用する。メッセージブローカとの通信には、OASIS によって標準化された MQTT プロトコルを採用し、誰でも利用可能なオープンな仕組みとする。

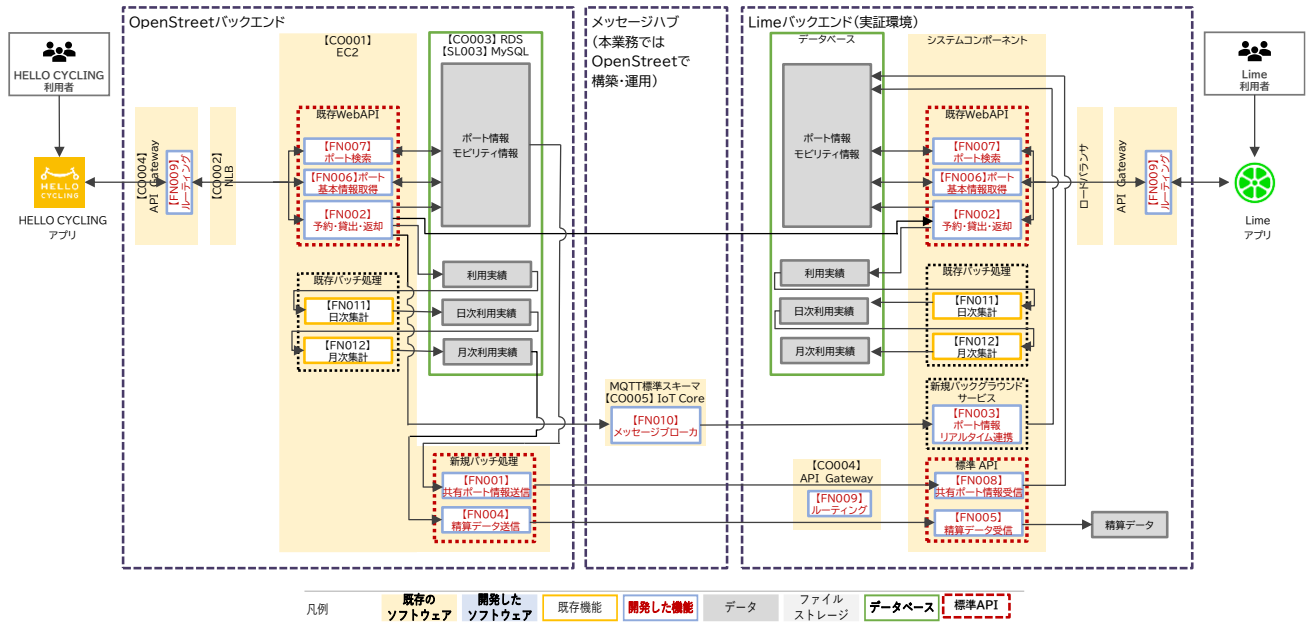


図 2-1 システムアーキテクチャ

2-1-2. システム機能一覧

表 2-1 システム機能一覧

※朱文字：新規開発・既存改修

ID	機能名	機能説明
FN001	共有ポート情報送信	● 標準 API を用いて、他の事業者へ自社の共有ポート情報を送信する
FN002	予約・貸出・返却	● 標準 API を用いて、予約・貸出・返却によりポートの空き状況が変化した際に、ポート情報リアルタイム連携の機能を呼び出す
FN003	ポート情報リアルタイム連携	● MQTT 標準スキーマを用いて、他事業者から受信した共同利用ポートの空き状況を自社のポート情報と連携する
FN004	精算データ送信	● 利用実績情報のうち、他事業者の共同利用ポートの利用実績情報から事業者間精算データを作成し、精算データ受信標準 API を介して送信する
FN005	精算データ受信	● 標準 API を用いて、他事業者から事業者間精算データを受信する
FN006	ポート基本情報取得	● 標準 API を用いて、自社及び他事業者のポートの静的な基本情報（名称、住所など）を取得する
FN007	ポート検索	● 標準 API を用いて、地図上又は現在地周辺における自社及び他事業者のポートを検索する
FN008	共有ポート情報受信	● 標準 API を用いて、他事業者から受信した共有ポート情報を自社情報に追加・更新・削除する
FN009	ルーティング	● リクエスト内容に応じて、適切な API にルーティングする
FN010	メッセージブローカ	● ポート情報リアルタイム連携でポート状態が変化した際に、それを通知するためのソフトウェア
FN011	日次集計	● 利用実績データを集計し、事業者別ポートの利用回数、利用時間などを日次単位で算出する
FN012	月次集計	● 利用実績データを集計し、事業者別ポートの利用回数、利用時間などを月次単位で算出する

2-1-3. システム機能の詳細

開発機能の詳細要件を記す。なお、本業務において、開発や改修を行う内容については、朱文字で示す。

【FN001】共有ポート情報送信＜新規開発＞

- 概要
  - 本機能は、他事業者から送信される共有ポート情報の取得リクエストを契機として、ポート ID、ポート名称、緯度・経度、最大駐輪可能台数などの情報を送信する
  - 本システムでは、共有ポート情報を一元管理せず、各事業者が他事業者のポート情報を自社ポートと同様に管理する
- 本システム機能の入力・処理・出力のフローチャート

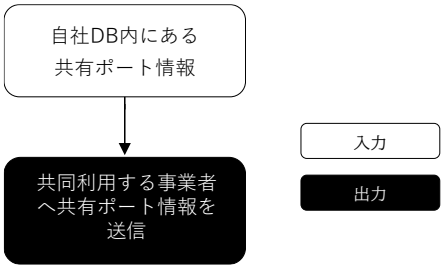


図 2-2 共有ポート情報送信のフローチャート

- 本システム機能の処理の詳細
  - 共有ポート情報の送信
    - 処理内容
      - ポートを共同利用する事業者へ共有ポート情報を送信する
    - 利用するライブラリ
      - 【SL001】OpenAPI
    - 利用するアルゴリズム
      - -
- データ仕様
  - 入力
    - 自社 DB 内の共有ポート情報
      - 内容
        - ステーション ID、緯度・経度、名称、最大収容数、リージョン ID、プロバイダー等
      - 形式
        - 自社 DB
      - 利用するデータインターフェース
        - 【IF001】共有ポート管理標準 API

- 出力
  - 他事業者へ送信する共有ポート情報
    - 内容
      - ポートを共同利用する事業者へ共有ポート情報を送信する
    - 形式
      - JSON
    - 利用するデータインターフェース
      - 【IF001】共有ポート管理標準 API

【FN002】予約・貸出・返却＜改修機能＞

- 概要
  - 本機能は、共同利用ポートにおいて予約・貸出・返却によりポートステータスが変化した際、ポートステータス情報を MQTT プロトコルでメッセージブローカへ送信し、他事業者にポートステータスの変化を通知する

既存機能の改修内容 *共有ポート情報および共有モビリティ情報の連携のため、自社ポートのみで予約・貸出・返却が可能だったものを、他事業者の共同利用ポートからの貸出や他事業者の共同利用ポートへの返却も可能となるように改修。モビリティの共同利用に対応するため、他事業者が管理するモビリティについても自社モビリティと同様に予約・貸出・返却の対象として扱い、モビリティステータスの変化を連携可能とするよう改修。ポートステータスの変更を他のシェアサイクル事業者に通知するように改修。*

- 本システム機能の入力・処理・出力のフローチャート

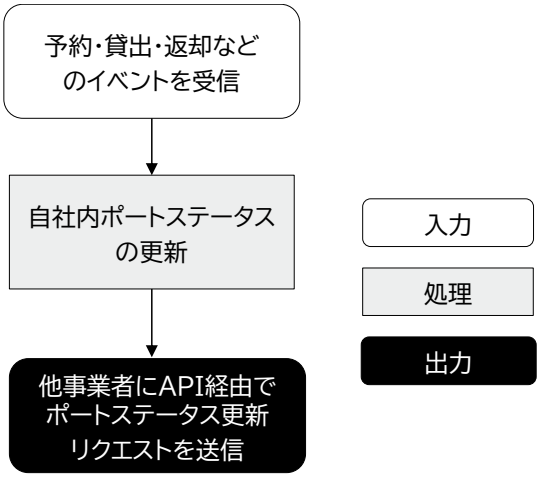


図 2-3 予約・貸出・返却のフローチャート（改修後）

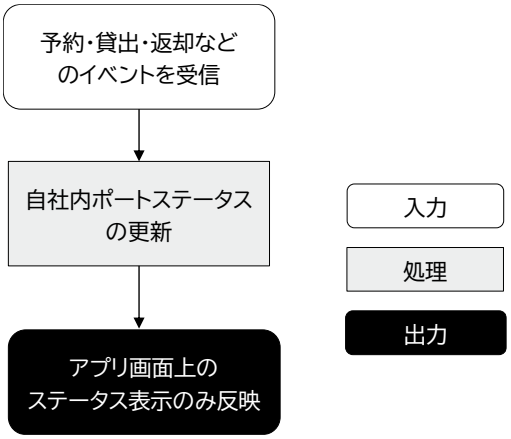


図 2-4 予約・貸出・返却のフローチャート（改修前）

- 本システム機能の処理の詳細
  - 予約・貸出・返却イベント処理
    - 処理内容
      - 予約・貸出・返却などのイベントを受信し、自社のポート情報を更新する
      - ポート情報リアルタイム連携 MQTT 標準スキーマを呼び出し、他事業者に対してリアルタイムでステータス情報を通知する
  - 利用するライブラリ
    - -
  - 利用するアルゴリズム
    - -
- データ仕様
  - 入力
    - 予約・貸出・返却イベント情報
      - 内容
        - ステーション ID、貸出可能台数、空きドック数など
      - 形式
        - JSON
      - 利用するデータインターフェース
        - 【IF008】解錠処理標準 API
        - 【IF009】返却処理標準 API
  - 出力
    - ポートステータスの更新通知
      - 内容
        - 予約・貸出・返却により変化したポートのステータスを他事業者に送信する
      - 形式
        - MQTT
      - 利用するデータインターフェース
        - 【IF002】ポート情報リアルタイム連携 MQTT 標準スキーマ

【FN003】ポート情報リアルタイム連携＜新規開発＞

- 概要
  - 本機能は、バックグラウンドサービスとして、メッセージブローカから MQTT プロトコルで送信される他事業者のポートステータスを常時監視し、メッセージを検知した場合、自社のポート管理データベースに反映する
  - 本システムでは、共有ポート情報を各事業者で管理するため、共同利用ポートの貸出可能台数や返却可能台数といったポートステータスを他事業者と同期する必要がある
- 本システム機能の入力・処理・出力のフローチャート

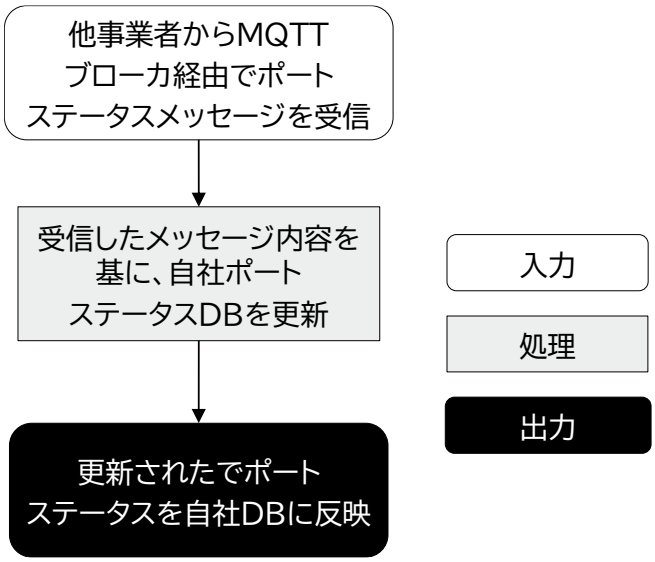


図 2-5 ポート情報リアルタイム連携のフローチャート

- 本システム機能の処理の詳細
  - リアルタイム情報の受信と反映
    - ポート情報リアルタイム連携を通じて他事業者から送られるステータス更新メッセージを受信する
    - リクエスト内容（ポート ID、貸出可能台数、空きドック数、更新時刻など）を基に、自社 DB のポート情報を更新する
  - 利用するライブラリ
    - -
  - 利用するアルゴリズム
    - -
- データ仕様
  - 入力
    - ポートステータス通知
      - 内容
        - ステーション ID、貸出可能台数、空きドック数など
      - 形式
        - MQTT

- 利用するデータインターフェース
  - 【IF002】ポート情報リアルタイム連携 MQTT 標準スキーマ

➢ 出力

✧ 更新後のポートステータス

- 内容
  - 自社 DB のポートステータス更新
- 形式
  - データベース更新
- 利用するデータインターフェース
  - 【IF002】ポート情報リアルタイム連携 MQTT 標準スキーマ

【FN004】精算データ送信＜新規開発＞

- 概要
  - ➢ 本機能は、日次又は月次で集計されたシェアサイクルの利用実績情報から共同利用ポートの利用実績情報を作成し、他事業者が公開している精算データ受信標準 API に送信する
  - ➢ 本機能により、自社の利用者が他事業者の管理する共同利用ポートを利用した場合、売上の一部を当該ポートを管理する事業者に支払うビジネスモデルを実現する
- 本システム機能の入力・処理・出力のフローチャート

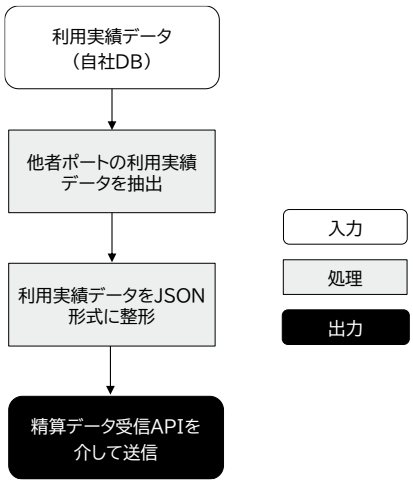


図 2-6 精算データ送信のフローチャート

- 本システム機能の処理の詳細
  - ➢ 利用実績データの集計・送信処理
    - ✧ 共同利用ポートの利用履歴から精算対象のデータを抽出
    - ✧ 各履歴に対し、注文 ID、貸出・返却ポート ID、開始・終了時刻、走行距離、利用料金などの項目を含め、事業者ごとに精算ファイルを生成
    - ✧ 他事業者の受信 API に対し、定期バッチで送信

- 利用するライブラリ
  - -
- 利用するアルゴリズム
  - -

- データ仕様
  - 入力
    - 利用実績データ
      - 内容
        - 自社 DB の共同利用ポートの利用履歴から抽出した精算対象のデータ（注文 ID、貸出・返却ポート ID、開始・終了時刻、走行距離、利用料金など）
      - 形式
        - 自社 DB
      - 利用するデータインターフェース
        - 【IF003】精算データ受信標準 API
  - 出力
    - 精算データ送信ファイル
      - 内容
        - 自社 DB から抽出した利用実績データ（注文 ID、貸出ポート ID、返却ポート ID、開始・終了時刻、走行距離、利用料金など）を、他事業者の精算データ受信標準 API へ送信する
      - 形式
        - JSON
      - 利用するデータインターフェース
        - 【IF003】精算データ受信標準 API

【FN005】精算データ受信＜新規開発＞

- 概要
  - 本機能は、他事業者から送信される自社管理の共同利用ポートに関する実績情報を受信し、事業者間精算における請求データを作成するために必要な情報として、自社のデータベースに登録する
  - 本機能により、自社の利用者が他事業者の管理する共同利用ポートを利用した場合、売上の一部を、当該ポートを管理する事業者に支払うビジネスモデルを実現する
- 本システム機能の入力・処理・出力のフローチャート

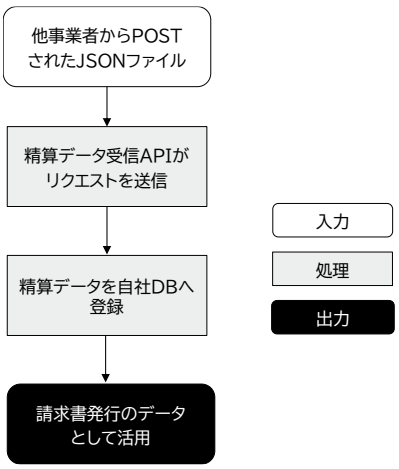


図 2-7 精算データ受信のフローチャート

- 本システム機能の処理の詳細
  - 精算データ受信・保存処理
    - 精算データ受信標準 API を介して POST された JSON 形式のデータを受信する
    - 自社 DB に登録する
  - 利用するライブラリ
    - 【SL001】OpenAPI
  - 利用するアルゴリズム
    - -
- データ仕様
  - 入力
    - 精算データファイル
      - 内容
        - 注文 ID、貸出・返却ポート ID、開始・終了時刻、走行距離、利用料金など
      - 形式
        - JSON
      - 利用するデータインターフェース
        - 【IF003】精算データ受信標準 API を参照
  - 出力
    - 自社 DB に登録した精算データ

- 内容
  - 請求書のデータとして活用される
- 形式
  - データベース更新
- 利用するデータインターフェース
  - 【IF003】精算データ受信標準 API を参照

【FN006】ポート基本情報取得＜改修機能＞

- 概要
  - 利用者がシェアサイクルアプリ上で特定のポートを選択した際、ポートの情報（ポート名称、住所、貸出可能台数、返却可能台数、ポート画像など）を表示するため、ポート ID を基に当該情報を取得する

既存機能の改修内容 *共有ポート情報の共有のため、自社ポートのみを対象としていた静的情報取得機能を、他事業者の共有ポート情報も取得可能とするように改修*

- 本システム機能の入力・処理・出力のフローチャート

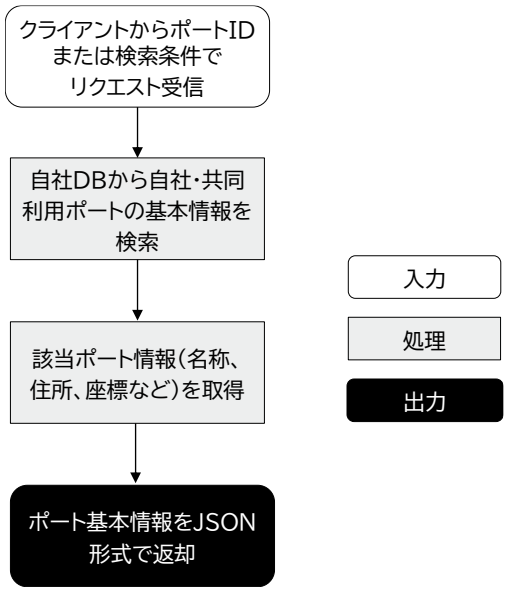


図 2-8 ポート基本情報取得のフローチャート（改修後）

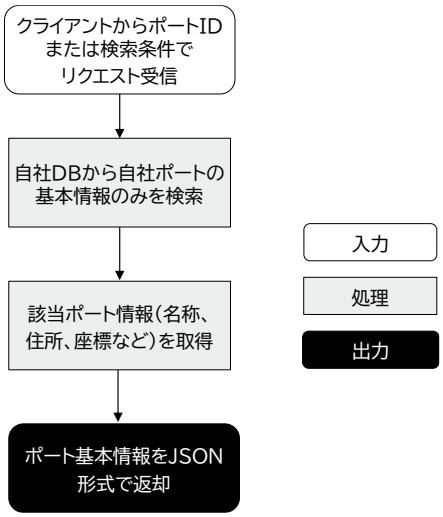


図 2-9 ポート基本情報取得のフローチャート（改修前）

- 本システム機能の処理の詳細
  - ポート基本情報の提供
    - 処理内容
      - シェアサイクルアプリや他事業者システムが API 経由で共有ポート情報を取得
      - 共有ポート情報は、共有ポート管理標準 API を通じて事前に取得・保持
    - 利用するライブラリ
      - -
    - 利用するアルゴリズム
      - -
- データ仕様
  - 入力
    - ポート ID（パラメータ）
      - 内容
        - 検索対象となるポート ID
      - 形式
        - GET リクエストパラメータ
      - 利用するデータインターフェース
        - 【IF004】ポート基本情報取得標準 API
  - 出力
    - ポート基本情報
      - 内容
        - ポート名称、座標、住所、最大収容台数、プロバイダー名など
      - 形式
        - JSON
      - 利用するデータインターフェース
        - 【IF004】ポート基本情報取得標準 API

【FN007】ポート検索＜改修機能＞

- 概要
  - 利用者がシェアサイクルアプリを起動した際又は周辺の貸出・返却可能なポートを確認する際に、Web 地図上でシェアサイクルポートの位置を表示するため、ポートのリストを取得する

既存機能の改修内容 *共有ポート情報の共有のため、自社ポートのみを対象としていた検索機能を、共同利用ポートも含めて検索可能となるように改修*

- 本システム機能の入力・処理・出力のフローチャート

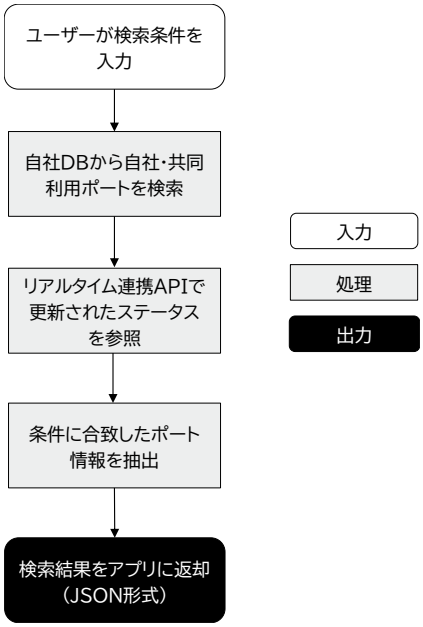


図 2-10 ポート検索のフローチャート（改修後）

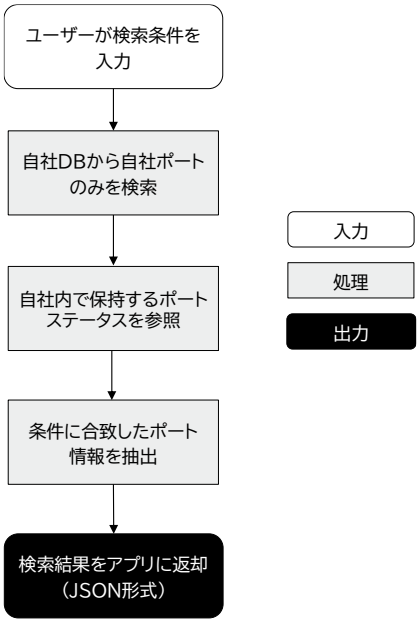


図 2-11 ポート検索のフローチャート（改修前）

- 本システム機能の処理の詳細
  - ポート検索
    - 処理内容
      - 位置情報や任意の検索条件（モビリティ種別など）を基に自社ポート・共同利用ポートを検索
      - ポートの空き状況（貸出可能台数、空きドック数など）は、ポート情報リアルタイム連携機能により更新されたデータを参照して取得
      - 結果をシェアサイクルアプリ向けに JSON 形式で返却
    - 利用するライブラリ
      - -
    - 利用するアルゴリズム
      - -
- データ仕様
  - 入力
    - 検索条件（JSON 又は GET パラメータ）
      - 内容
        - 現在地座標、検索範囲、対応車種など
      - 形式
        - GET リクエストパラメータ
      - 利用するデータインターフェース
        - 【IF005】ポート検索標準 API
  - 出力
    - ポート検索結果（JSON）
      - 内容
        - ポート ID、名称、住所、座標、貸出可能台数、空きドック数、対応モビリティなど
      - 形式
        - JSON
      - 利用するデータインターフェース
        - 【IF005】ポート検索標準 API

【FN008】共有ポート情報受信＜新規開発＞

- 概要
  - 本機能は、共同利用する他事業者が公開する API を定期的に実行し、ポート名称、住所、緯度・経度などの情報を取得し、自社のポート管理データベースに登録する
  - その際、各事業者のポート ID が重複しないよう、一意な ID を発番する
  - 本システムでは、共有ポート情報を一元管理するのではなく、各事業者が他事業者の共有ポート情報を自社ポートと同様に管理する
- 本システム機能の入力・処理・出力のフローチャート

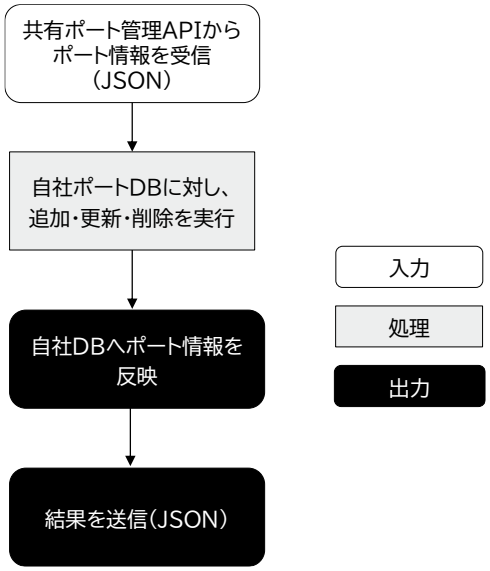


図 2-12 共有ポート情報受信のフローチャート

- 本システム機能の処理の詳細
  - 共有ポート情報の受信と自社 DB への登録
    - 処理内容
      - ポートを共同利用する事業者から共有ポート情報を受信する
      - 自社 DB に対して共有ポート情報を登録する
    - 利用するライブラリ
      - 【SL001】OpenAPI
    - 利用するアルゴリズム
      - -
- データ仕様
  - 入力
    - 共有ポート情報
      - 内容
        - 他事業者から標準 API を介して受信した共有ポート情報（ステーション ID、緯度・経度、名称、最大収容数、プロバイダー等）
      - 形式
        - JSON
      - 利用するデータインターフェース
        - 【IF001】共有ポート管理標準 API
  - 出力
    - 更新後の共有ポート情報
      - 内容
        - 受信した共有ポート情報（ステーション ID、緯度・経度、名称、最大収容数、プロバ

イダー等）を自社 DB に登録する

- 形式
  - データベース更新、レスポンス（JSON）
- 利用するデータインターフェース
  - 【IF001】共有ポート管理標準 API

【FN009】ルーティング＜改修機能＞

- 概要
  - 本機能は、シェアサイクルアプリやポートを共同利用する他事業者から送信される API リクエストを受信し、HTTP メソッド及びエンドポイントに基づき、対応するバックエンドの処理を呼び出す

既存機能の改修内容
既存のポート検索、ポート基本情報取得、予約・貸出・返却といった API リクエストに加え、共有ポート情報送信や精算データ受信のルーティングに対応するように改修。

- 本システム機能の入力・処理・出力のフローチャート

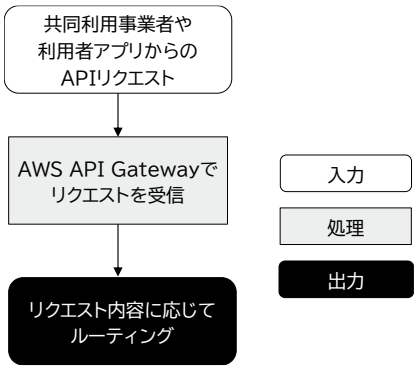


図 2-13 ルーティングのフローチャート

- 本システム機能の処理の詳細
  - ✧ ルーティング
    - APIGateway でリクエストを受信し、ルーティングする
  - ✧ 利用するライブラリ
    - -
  - ✧ 利用するアルゴリズム
- -データ仕様
  - 入力
    - ✧ 他事業者やシェアサイクルアプリからの API リクエスト
      - 内容
        - 他事業者やシェアサイクルアプリから送信されるリクエスト
      - 形式
        - JSON
      - 利用するデータインターフェース

- 【IF001】共有ポート管理標準 API
- 【IF002】ポート情報リアルタイム連携 MQTT 標準スキーマ
- 【IF003】精算データ受信標準 API

- 出力
  - ルーティング先の EC2 群
    - 内容
      - 受信したリクエストを対応するバックエンド処理へ転送し、共有ポート管理標準 API であればポート基本情報、ポート情報リアルタイム連携であればポートステータス情報、精算データ受信標準 API であれば利用実績情報として、処理結果を DB に反映する
    - 形式
      - JSON
    - 利用するデータインターフェース
      - 【IF001】共有ポート管理標準 API
      - 【IF002】ポート情報リアルタイム連携 MQTT 標準スキーマ
      - 【IF003】精算データ受信標準 API

【FN010】メッセージブローカ＜新規開発＞

- 概要
  - 本機能では、共同利用ポートの貸出可能台数や返却可能台数などのポートステータスが変化した際、メッセージを受信し、ポートを共同利用する他事業者に一斉に配信する
  - 本システムでは、3 社以上によるポートの共同利用を見据え、N 対 N の情報連携を柔軟に実現可能なアーキテクチャとして、メッセージブローカを介してポートステータスを共有する Pub/Sub 型アーキテクチャを採用した
- 本システム機能の入力・処理・出力のフローチャート

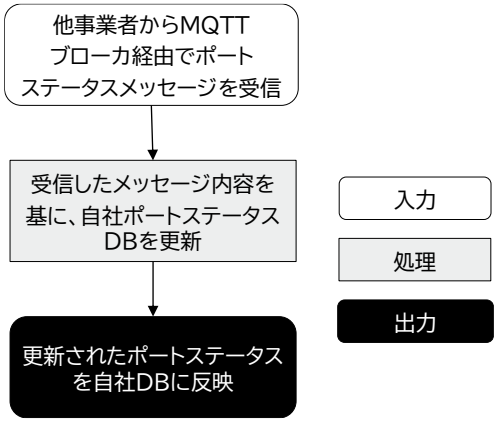


図 2-14 MQTT のフローチャート

- 本システム機能の処理の詳細
  - ポートステータス情報の事業者間
    - 自社 DB でポート状態を検知した場合に、ステータス更新メッセージを MQTT で送信する
    - 他事業者は当該メッセージを受信し、自社の DB を更新する
  - 利用するライブラリ
    - -
  - 利用するアルゴリズム
    - -
- データ仕様
  - 入力
    - 自社 DB から検知されたポート状態
      - 内容
        - ポートステータス情報
      - 形式
        - 自社 DB
      - 利用するデータインターフェース
        - 【IF002】ポート情報リアルタイム連携 MQTT 標準スキーマ
  - 出力
    - ポートステータス通知
      - 内容
        - ポートステータス情報
      - 形式
        - MQTT
      - 利用するデータインターフェース
        - 【IF002】ポート情報リアルタイム連携 MQTT 標準スキーマ

【FN011】日次集計＜新規開発＞

- 概要
  - サービスの改善及び円滑な事業運営のため、シェアサイクルの利用履歴情報から、貸出ポート、返却ポート、利用時間、利用回数などの項目で日次集計を実施し、利用実績情報を作成する
- 本システム機能の入力・処理・出力のフローチャート

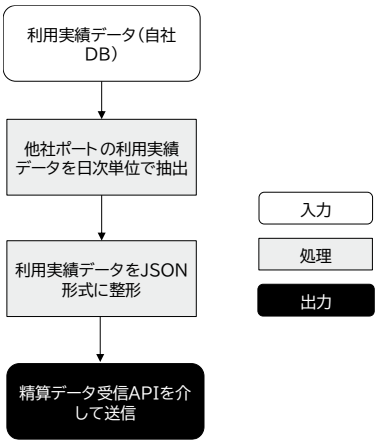


図 2-15 日次集計のフローチャート

- 本システム機能の処理の詳細
  - 日次集計処理
    - 利用実績データを抽出し、事業者別のポート利用回数、利用時間などを集計
    - 集計結果を他事業者へ送信する
  - 利用するライブラリ
    - -
  - 利用するアルゴリズム
    - -
- データ仕様
  - 入力
    - 日次集計処理
      - 内容
        - 利用開始時刻、終了時刻、利用時間、利用料金、利用ポート情報など
        - 集計結果を他事業者へ送信する
      - 形式
        - JSON
      - 利用するデータインターフェース
        - 【IF003】精算データ受信標準 API
  - 出力
    - 日次集計結果
      - 内容
        - 他事業者から受信した日次利用実績データから自社ポートの利用開始時刻、終了時刻、

利用時間、利用料金、利用ポート情報などを自社 DB に登録

- 形式
  - 自社 DB
- 利用するデータインターフェース
  - 【IF003】精算データ受信標準 API

【FN012】月次集計＜新規開発＞

- 概要
  - サービスの改善及び円滑な事業運営並びに協定を締結している地方公共団体への報告のため、日次で集計された利用実績情報から月次の利用実績情報を作成する
- 本システム機能の入力・処理・出力のフローチャート

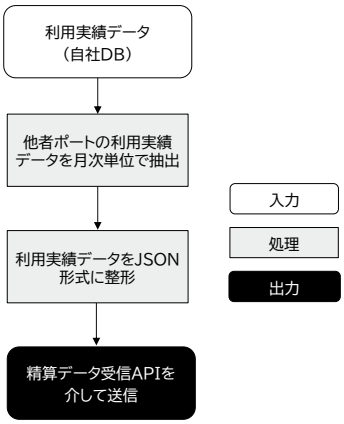


図 2-16 月次集計のフローチャート

- 本システム機能の処理の詳細
  - ✧ 月次集計処理
    - 1 か月分の利用実績データを抽出し、事業者別の利用回数、利用時間、収益を集計
    - 集計結果を他事業者へ送信する
  - ✧ 利用するライブラリ
    - -
  - ✧ 利用するアルゴリズム
    - -
- データ仕様
  - 入力
    - ✧ 月次集計処理
      - 内容
        - 利用開始時刻、終了時刻、利用時間、利用料金、利用ポート情報など
        - 集計結果を他事業者へ送信する
      - 形式
        - JSON
      - 利用するデータインターフェース

- 【IF003】精算データ受信標準 API

- 出力
  - 月次集計結果
    - 内容
      - 他事業者から受信した月次利用実績データから自社ポートの利用開始時刻、終了時刻、利用時間、利用料金、利用ポート情報などを自社 DB に登録
    - 形式
      - -
    - 利用するデータインターフェース
      - 【IF003】精算データ受信標準 API

2-1-4. ソフトウェア・ライブラリ（SL）の詳細

表 2-2 ソフトウェア・ライブラリ一覧

※朱文字：新規開発・既存改修

ID	名称	バージョン	内容
SL001	OpenAPI	3.0.3	● REST API の仕様を記述するためのフレームワーク
SL002	Laravel	-	● PHP ベースの Web アプリケーションフレームワーク
SL003	MySQL	-	● データベース

システムコンポーネントの詳細を記す。なお、本業務において開発（新規・改修）を行うシステムコンポーネントを朱文字で示す。

【SL001】OpenAPI

- ベンダー
  - OpenAPI Initiative（Linux Foundation）
- 公式サイト
  - https://www.openapis.org/
- 概要
  - OpenAPI は、REST API の設計・定義・共有のための国際的な標準仕様であり、YAML 又は JSON 形式で API 仕様を記述できる
- 主な機能
  - API 仕様の文書化
  - リクエスト／レスポンスのスキーマ定義
  - Swagger UI によるインタラクティブ API ドキュメント生成
- イメージ
  - -

【SL002】Laravel

- ベンダー
  - Laravel LLC
- 公式サイト
  - https://laravel.com/
- 概要
  - PHP で書かれたオープンソースの Web アプリケーションフレームワーク
  - REST API の構築にも広く用いられる
- 主な機能
  - ルーティング、ミドルウェア
  - Eloquent ORM による DB 操作の簡略化
  - Artisan CLI によるマイグレーション管理
- イメージ
  - -

【SL003】MySQL

- ベンダー
  - Oracle Corporation
- 公式サイト
  - https://www.mysql.com/
- 概要
  - 世界で広く利用されているオープンソースのリレーショナルデータベース管理システム
- 主な機能
  - データの構造化保存（テーブル、インデックス、外部キーなど）
  - SQL によるデータ検索・集計・加工処理
- イメージ
  - -

2-1-5. 数理モデル・アルゴリズム（AL）の詳細

-

2-2. システムコンポーネント（CO）

2-2-1. システムコンポーネント図

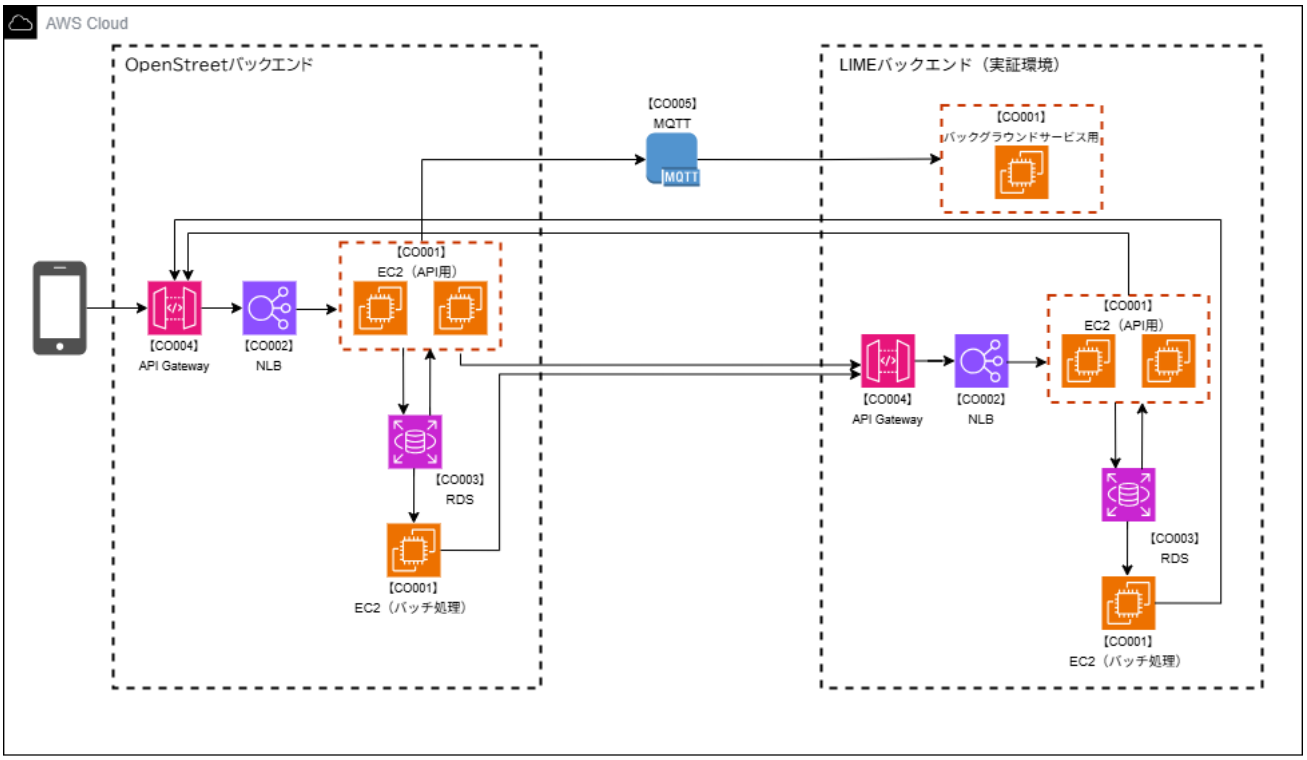


図 2-17 システムコンポーネント図

2-2-2. システムコンポーネント一覧

表 2-3 システムコンポーネント一覧

ID	種別	コンポーネント名	用途
CO001	IaaS	EC2	● API やバッチ処理を実行する仮想サーバ
CO002	IaaS	NLB	● EC2 インスタンスへの負荷分散
CO003	PaaS	RDS	● データベース
CO004	PaaS	API Gateway	● API リクエストのルーティング・認証
CO005	PaaS	MQTT	● ポートステータスのメッセージ配信

2-3. ハードウェア（HW）

2-3-1. ハードウェアアーキテクチャ

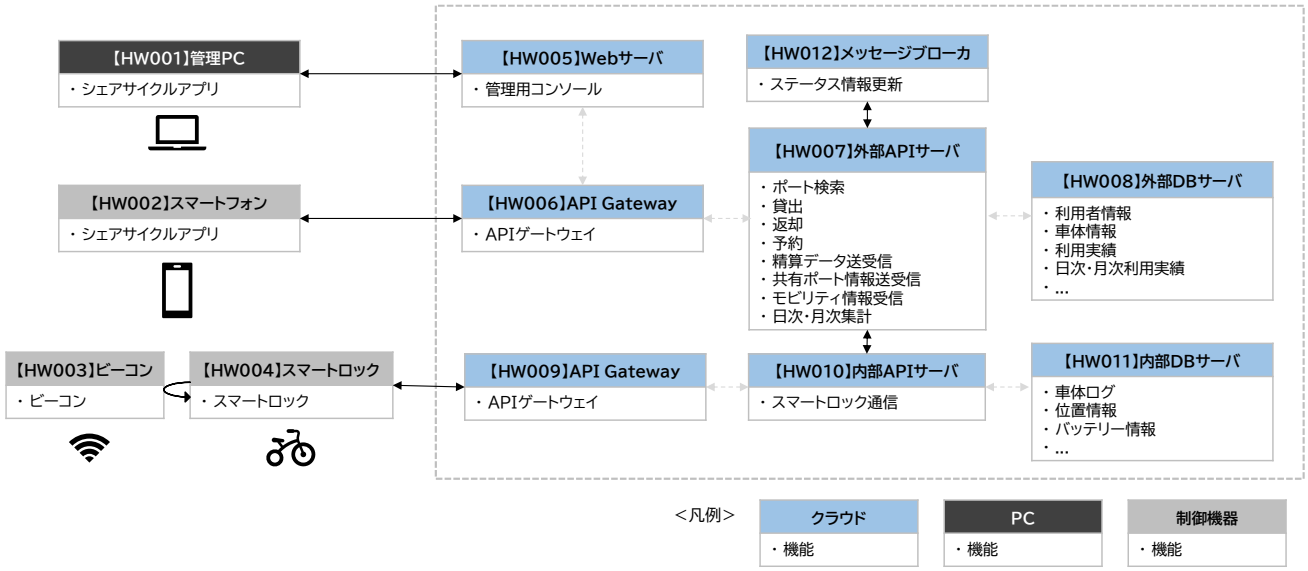


図 2-18 ハードウェアアーキテクチャ

2-3-2. ハードウェア一覧

表 2-4 ハードウェア一覧

※朱文字：新規開発・既存改修

ID	種別	ベンダー	品番	用途
HW001	管理 PC	-	-	● 管理コンソールクライアント
HW002	スマートフォン	Android/iOS	-	● シェアサイクルアプリ
HW003	ビーコン	Raytac	MDBT42Q-192KV2	● 返却時にどのステーションに返却しようとしているかを判別 ● 事業者によって違いはあるが、ビーコンと GPS を併用していることが多い
HW004	スマートロック	OpenStreet	HC0065	● 自転車に取り付けられているスマートロック。SIM が入っており通信ができるようになっている
HW005	Web サーバ	AWS EC2	-	● 管理コンソール用の Web サーバ
HW006	API ゲートウェイ（外部）	AWS API Gateway	-	● シェアサイクルアプリとバックエンドをつなぐゲートウェイ
HW007	API サーバ（外部）	AWS EC2	-	● シェアサイクルアプリ向け API サーバ
HW008	DB サーバ（外部）	Amazon RDS	-	● シェアサイクルアプリ向け DB サーバ
HW009	API ゲートウェイ（内部）	AWS API Gateway	-	● スマートロックとバックエンドをつなぐゲートウェイ
HW010	API サーバ（内部）	AWS EC2	-	● スマートロック通信用の API サーバ
HW011	DB サーバ（内部）	Amazon RDS	-	● スマートロック通信用の DB サーバ
HW012	メッセージブローカ	AWS IoT Core	-	● ポート状態が変化した際に通知するためのメッセージブローカ

2-3-3. ハードウェアの詳細

ハードウェアの詳細を記す。なお、本業務において開発（新規・改修）を行うハードウェアを朱文字で示す。

【HW001】管理 PC

- 概要
  - シェアサイクル事業の運営管理を行うための Web ベースの管理システム
- ベンダー
  - OpenStreet 株式会社が自社で開発・保守
- 仕様・スペック
  - クラウド上で提供される Web アプリケーションであり、専用ソフトウェアのインストールは不要
  - Google Chrome の最新版を推奨し、PC のスペックは定めなし。上記ブラウザが快適に動作する一般的な性能の PC（OS は Windows/macOS を問わない）
- イメージ
  - -

【HW002】スマートフォン

- 概要
  - シェアサイクルアプリを利用するのに十分なスペックを持つスマートフォン
- ベンダー
  - Android 又は iOS
- 仕様・スペック
  - シェアサイクルアプリの動作環境：Android 10 以降、 iOS 16.0 以降
- イメージ
  - -

【HW003】ビーコン

- 概要
  - 自転車返却時に、返却するステーションを判別するために使用される位置検知デバイス
  - 事業者によって違いはあるが、ビーコンと GPS を併用していることが多い
- ベンダー
  - Raytac
- 仕様・スペック
  - 搭載 SoC：Nordic Semiconductor 社製 nRF52810 SoC
  - Bluetooth 規格：BT5.2/BT5.1/BT5/BT4.2 認証済み
  - メモリ：192KB Flash Memory
  - RAM：24KB
  - アンテナ：Chip Antenna

- イメージ
  - -

【HW004】スマートロック

- 概要
  - 自転車に取り付いている通信機能付きスマートロック
  - 利用者の解錠操作に応じて、サーバと通信し、利用開始のステータスを管理する
- ベンダー
  - OpenStreet 株式会社
- 仕様・スペック
  - 入力電圧：DC7.5V ～ DC42.0V
  - コネクタ：68R-JWPF-VSLE-D（8pin 電源ケーブル・防水）
  - 動作温度・湿度範囲：-20℃ ～ 60℃、20% ～ 80%（結露なきこと）
  - サイズ：L194 × W148 × H41mm（ハーネス除く）
  - ケース材質：PC＋10Kgf（TOP）、Aluminum alloy（BOTTOM）
  - 重量：約 720g
  - 耐振動性：JIS C 60068-2-27 相当
  - 防水防塵等級：IPX6 相当
  - 保証期間：製品アクティベーション後 1 年間
  - 消費電力：待機中 424mW / 動作中 490mW
- イメージ

図 2-19 スマートロック

【HW005】Web サーバ

- 概要
  - 管理コンソール用の Web サーバ
- ベンダー

- AWS EC2
- 仕様・スペック
  - -
- イメージ
  - -

【HW006】API ゲートウェイ（外部）〈既存改修〉

- 概要
  - 他事業者との通信を中継し、新規開発 API に対してルーティングを行う
- ベンダー
  - AWS API Gateway
- 仕様・スペック
  - 認証方式： OAuth2.0
- イメージ
  - -

【HW007】API サーバ（外部）〈既存改修〉

- 概要
  - 外部システムからのリクエストに対応する処理ロジックを保持する
- ベンダー
  - AWS EC2
- 仕様・スペック
  - Laravel（PHP フレームワーク）
- イメージ
  - -

【HW008】DB サーバ（外部）〈既存改修〉

- 概要
  - 共同利用ポート情報、利用情報、精算情報などを保存・管理するデータベース
- ベンダー
  - Amazon RDS
- 仕様・スペック
  - MySQL
- イメージ
  - -

【HW009】API ゲートウェイ（内部）

- 概要

    - スマートロックとバックエンドシステム間の通信を中継するゲートウェイ
- ベンダー
    - AWS API Gateway
- 仕様・スペック
    - -
- イメージ
    - -

【HW010】API サーバ（内部）

- 概要
    - スマートロックからの通信を受け取り、解錠処理やステータス管理を行う API サーバ
- ベンダー
    - AWS EC2
- 仕様・スペック
    - -
- イメージ
    - -

【HW011】DB サーバ（内部）

- 概要
    - スマートロックの状態、GPS 情報などを保存・管理する内部データベース
- ベンダー
    - Amazon RDS
- 仕様・スペック
    - -
- イメ-ジ
    - -

【HW012】メッセージブローカ〈新規開発〉

- 概要
    - ポート情報リアルタイム連携でポート状態が変化した際に通知するためのメッセージブローカ
- ベンダー
    - AWS IoT Core
- 仕様・スペック
    - -
- イメージ
    - -

2-4. データインターフェース（IF）

2-4-1. データアーキテクチャ

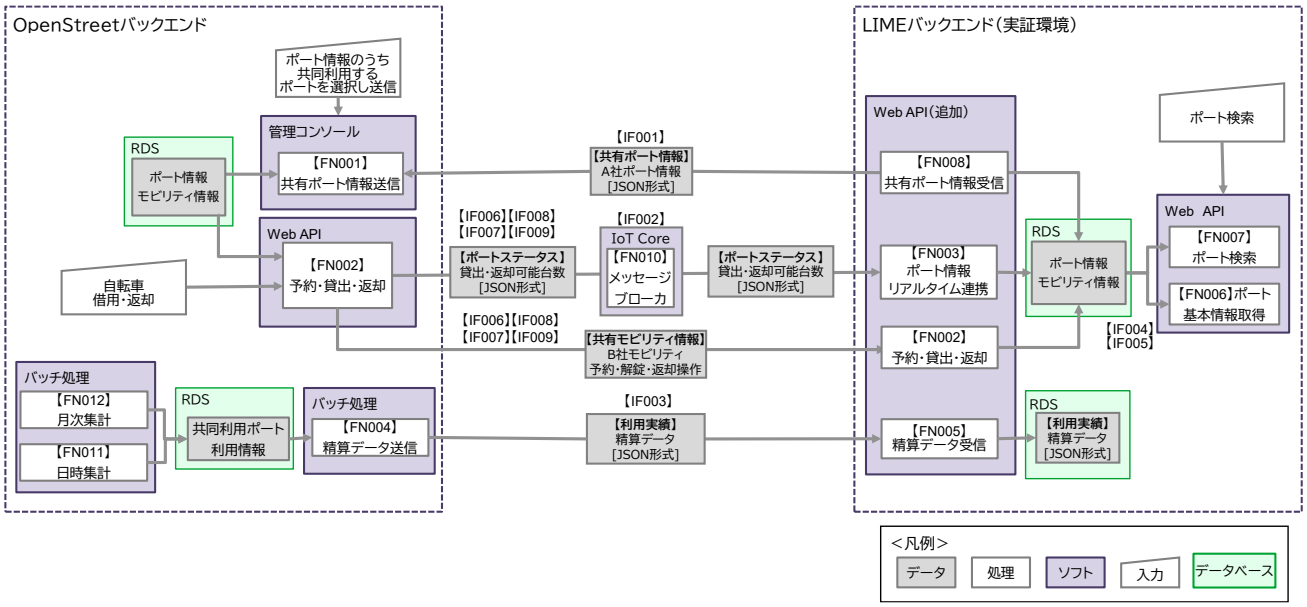


図 2-20 データアーキテクチャ

2-4-2. データインターフェース一覧

表 2-5 データインターフェース一覧

※朱文字：新規開発・既存改修

ID	名称	出力側 ID	入力側 ID
IF001	共有ポート管理標準 API	FN008	FN001
IF002	ポート情報リアルタイム連携 MQTT 標準スキーマ	FN002	FN003
IF003	精算データ受信標準 API	FN005	FN004
IF004	ポート基本情報取得標準 API	FN006	FN006
IF005	ポート検索標準 API	FN007	FN007
IF006	予約登録標準 API	FN003	FN002
IF007	予約キャンセル標準 API	FN003	FN002
IF008	解錠処理標準 API	FN003	FN002
IF009	返却処理標準 API	FN003	FN002

2-4-3. データインターフェースの詳細

データインターフェースの詳細を記す。なお、本業務において開発（新規・改修）を行うデータインターフェースを朱文字で示す。

【IF001】共有ポート管理標準 API＜新規開発＞

- 概要
  - ポートを共同利用する他事業者から共有ポート情報取得リクエストを受信し、ポート ID、ポート名称、緯度・経度、最大駐輪可能台数、ポート画像 URL などの情報を送信する
- 本インターフェースを利用する機能
  - 【FN001】共有ポート情報送信
  - 【FN008】共有ポート情報受信
- 共通仕様
  - プロトコル
    - HTTPS
  - メソッド
    - GET
  - リクエスト形式
    - HTTP ヘッダ
  - レスポンス形式
    - JSON
  - 文字コード
    - UTF-8
  - リクエストヘッダ

UTF-8 名称	説明	型	必須
ak	アクセスキー	文字列	○
rq	任意の数値を head 内にそのまま返却（識別用ヘッダパラメータ）	文字列	-

  - リクエストパラメータ

項目	名称	必須	データ型	説明	値
データ取得対象日時	since	○	datetime	指定した日時より後に追加・更新されたポートの情報	例：2023-10-27T10:30:00Z

➢ レスポンスパラメータ

項目	名称	必須	データ型	説明	値
企業 ID	id	○	String	サービを提供する企業（HC、Lime）を一意に識別するための ID	例：os
企業名	name	○	Array	サービスを提供している事業者の名称	-
言語コード	language	○	String	言語コード	例：JA
企業名	text	○	String	企業名	例：OpenStreet 株式会社
ブランド名	brand_name	-	String	サービスのブランド名	例：HELLO CYCLING
ブランドのロゴ画像 URL	brand_image_url	-	String	地図上に表示するための、ブランドを識別できるロゴ画像の URL	例：https://d2a2mitt2l0zmn.cloudfront.net/data/2019/05/07/keep_aspect_ratio/120/HC-logo.png
ポート情報	stations	○	Array	ポート情報の一覧	-
ポート ID	id	○	String	ポートを一意に識別するための ID	例：6113
ポート名	name	○	Array	利用者に表示されるポートの名称	-
言語コード	language	○	String	言語コード	例：JA
企業名	text	○	String	企業名	例：竜泉 1 丁目
緯度	lat	○	Number	ポートが設置されている地点の緯度	例：35.724652
経度	lon	○	Number	ポートが設置されている地点の経度 世界測地系（緯度・経度座標系）WGS84	例：139.789753
住所	address	○	String	ポートの所在地	例：東京都台東区竜泉 1-33-7
最大駐輪可能台数	capacity	○	Integer	ポートに駐輪可能な最大台数	例：6
ポート画像 URL	image_url	-	String	ポートの外観が分かる画像の URL	例：https://d2a2mitt2l0zmn.cloudfront.net/

					data/2019/03/01/keep_aspect_ratio/750/FF2A6E0D-B934-4329-934C-0D5DACE717F9.jpeg

➢ レスポンスサンプル


```
{
  "id": “HC”,
  "name": [
    {
      "language": "JA",
      "text": "OpenStreet 株式会社"
    }
  ],
  "brand_name": "HELLO CYCLING",
  " brand_image_url":
"https://d2a2mitt2l0zmn.cloudfront.net/data/2019/05/07/keep_aspect_ratio/120/HC-logo.png",
  "stations": [
    {
      "id": 6113,
      "name": [
        {
          "language": "JA",
          "text": "竜泉 1 丁目"
        }
      ],
      "lat": 35.724652,
      "lon": 139.789753,
      "address": "東京都台東区竜泉 1-33-7",
      "capacity": 11,
      "image_url": "https://d2a2mitt2l0zmn.cloudfront.net/data/2022/07/11/3447.jpg"
    }
  ]
}
```

【IF002】ポート情報リアルタイム連携 MQTT 標準スキーマ＜新規開発＞

- 概要
  - 共同利用ポートの貸出可能台数や返却可能台数などのポートステータスが変化した際、変化後のポートステータスをメッセージで受信し、当該メッセージを、ポートを共同利用する他事業者に一斉に配信する
- 本インターフェースを利用する機能
  - 【FN002】予約・貸出・返却
  - 【FN003】ポート情報リアルタイム連携
- 共通仕様
  - プロトコル
    - MQTT
  - 通信様式
    - Publish/Subscribe
  - ペイロード
    - UTF-8/JSON
  - トピック名称
    - jp/events/{source_partner_id}/stations/{station_id}/status
  - ペイロード仕様

```
{
  "event_id": "evt_1M5xYzE4g",
  "event_type": "port.status_updated",
  "source_partner_id": "HC",
  "timestamp": "2025-10-21T10:30:00Z",
  "payload": {
    "station_id": 1234,
    "num_vehicles_available": 5,
    "num_docks_available": 10,
    "reason_code": "VEHICLE_RETURNED"
  }
}
```

- サンプルコード

```
import paho.mqtt.client as mqtt
import json

# 接続時に呼び出されるコールバック
def on_connect(client, userdata, flags, rc):
    print("MQTT ブローカに接続しました。")
    # OpenStreet 社の全ポートのステータスイベントを購読
    topic = "jp/events/HC/stations/+/status"
    print(f"トピック '{topic}' を購読します。")
    client.subscribe(topic)

# メッセージ受信時に呼び出されるコールバック
def on_message(client, userdata, msg):
    try:
        payload_data = json.loads(msg.payload.decode())
        station_id = payload_data.get("payload", {}).get("station_id")
        num_vehicles_available = payload_data.get("payload", {}).get("num_vehicles_available")
        docks = payload_data.get("payload", {}).get("num_docks_available")
        print(f"ポート ID {station_id} の情報が更新されました: 貸出可能={vehicles}台, 返却可能={docks}台")
    except (json.JSONDecodeError, AttributeError):
        print(f"エラー: 受信したメッセージの形式が正しくありません: {msg.payload.decode()}")

# MQTT クライアントのセットアップ
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

# ブローカに接続（アドレス、ポートは実証実験環境のものに要変更）
try:
    client.connect("mqtt.example.com", 1883, 60)
    client.loop_forever() # 永久ループでメッセージを待ち受ける
except Exception as e:
    print(f"接続に失敗しました: {e}")
```

【IF003】精算データ受信標準 API＜新規開発＞

- 概要
  - ポートを共同利用する事業者から自社の共同利用ポートの利用実績データを受信し、事業者間精算のための情報とするため、自社のデータベースに登録し、登録結果を返信する。
- 本インターフェースを利用する機能
  - 【FN004】精算データ送信
  - 【FN005】精算データ受信
- 共通仕様
  - プロトコル
    - HTTPS
  - メソッド
    - POST
  - リクエスト形式
    - JSON
  - レスポンス形式
    - JSON
  - 文字コード
    - UTF-8

  - リクエストヘッダ

x-api-key: （正しい値）

  - リクエストパラメータ

項目	名称	必須	データ型	説明	値
ファイル全体に関するメタ情報を含む JSON 文字列	payload	○	String (JSON)	精算対象月など、ファイル全体に関するメタ情報を含む JSON 文字列	例：{"settlement_month":"2025-09"}
精算対象の利用実績データ	transactions_file	○	File	精算対象の利用実績データ（JSON 形式）を含むファイル本体	例：transactions_202509.json

➢ リクエストボディサンプル

```
{
  "payload": "{ "settlement_month": "2025-09"}",
  "transactions_file": "transactions_202509.json"
}
```

➢ レスポンスヘッダ

HTTP ステータスコード: 200

➢ レスポンスパラメータ

項目	名称	必須	データ型	説明	値
レスポンスデータ	data	○	Object	レスポンスデータ	-
バッチ ID	batch_id	○	Integer	受理されたバッチの ID	例：BCH_asdf12345678
ステータス	status	○	String	現在の受付状態	例：received
メッセージ	message	-	String	メッセージ	例：精算データファイルを受理しました。

➢ レスポンスサンプル

```
    "data": [
      {
        "batch_id": "BCH_asdf12345678",
        "status": "received",
        "message": "精算データファイルを受理しました"
      }
    ]
  }
}
```

【IF004】ポート基本情報取得標準 API＜既存改修＞

- 概要
  - ➢ ポート ID を基に、ポート名称、緯度・経度、住所、利用可能な自転車情報、ポートの写真画像 URL などの情報を送信する
- 本インターフェースを利用する機能

- 【FN006】ポート基本情報取得

- 共通仕様
  - プロトコル
    - HTTPS
  - メソッド
    - GET
  - リクエスト形式
    - クエリ
  - レスポンス形式
    - JSON
  - 文字コード
    - UTF-8
  - リクエストクエリ

項目	名称	必須	データ型	説明	値
ポート ID	station_id	○	String	ポートを一意に識別するための ID	例：6113
認証トークン	rest_token	-	String	認証トークン	例：7c74bcde4f03db34af7acbbe7f6b321d

  - レスポンスパラメータ

項目	名称	必須	データ型	説明	値
レスポンスデータ	data	○	Object	レスポンスデータ	-
ポート ID	id	○	String	ポートを一意に識別するための ID	例：6113
ポート名	name	○	Array	利用者向けに表示されるポートの名称	-
言語コード	language	○	String	言語コード	JA
ポート名	text	○	String	ポート名	竜泉 1 丁目
ブランド名	brand_name	-	String	ブランド名	例：HELLO CYCLING
企業名	name	○	Array	サービスを提供している事業者の名称	-
言語コード	language	○	String	言語コード	例：JA
企業名	text	○	String	企業名	例：OpenStreet 株式会社
ブランド公式	url	-	String	運営会社の詳細情報を	例：

URL				確認できる公式 Web ページの URL	https://resource.hellocycling.jp/webview/company/detail/11
会社 ID	system_id	-	String	運営会社を一意に識別するための ID	例：11
ブランドロゴ画像 URL	brand_image_url	-	String	ブランドを識別するためのロゴ画像の URL	例：https://d2a2mitt2l0zmn.cloudfront.net/data/2019/05/07/keep_aspect_ratio/120/HC-logo.png
住所	address	○	String	ポートが設置されている所在地の住所表記	東京都台東区竜泉 1-33-7
開店時間	opening_hour	-	String	ポートの営業開始時刻	例：00:00
営業時間	station_opening_hours	-	String	ポートの営業時間	例：24h
駐輪制限有無	has_parking_limit	○	Boolean	ポートに駐輪制限が設定されているかどうかを示すフラグ	例：TRUE
駐輪可能台数	num_docks_available	○	Integer	当該ポートに駐輪可能な自転車の台数	例：6
貸出可能台数	num_vehicles_available	○	Integer	当該ポートにおいて現在貸出可能な状態にある自転車の台数	例：3
予約済み自転車台数	num_vehicles_reserved	○	Integer	当該ポートにおいてすでに予約されている自転車の台数	例：0
緯度	lat	○	Number	ポートが設置されている地点の緯度	例：35.724652
経度	lon	○	Number	ポートが設置されている地点の経度	例：139.789753
バッテリーフ	is_charging_statio	○	Boolean	バッテリー状態を示す	例：TRUE

ラグ	n			フラグ	
郵便番号	post_code	○	String	ポート所在地の郵便番号	例：110-0012
自転車情報一覧	vehicles	○	Array	自転車情報の一覧	-
自転車 ID	id	○	String	自転車を一意に識別するための ID	例：20105
カテゴリ ID	vehicle_type_id	○	String	自転車のカテゴリ ID	2：電動アシスト(EV_ASSIST_BIKE)、3：デモ機(DEMO_BIKE)、4：KUROAD 車両(KUROAD_BIKE)、5：一般自転車(GENERAL_BIKE)、6：スポーツタイプ(SPORT_BIKE)、13：チャイルドシート付き車両(CHILD_SHEET_BIKE)、14：ベロスター(VELOSTAR)、15：電動サイクル(ELECTRIC_BIKE)、16：DBS 車両(DBS_BIKE)

					、17：glafit 車両 (GLAFIT)
単位料金	price	-	Number	時間や距離単位で加算される料金設定	例：160
時間料金	hour_price	-	Number	時間貸し利用時の料金設定	例：0
1 日料金	day_price	-	Number	1 日単位で利用する場合の料金設定	-
半日料金	half_day_price	-	Number	半日単位で利用する場合の料金設定	例：2500
自転車写真 URL	photo_path	-	String	自転車の外観が分かる写真画像の URL	例：https://d2a2mitt2l0zmn.cloudfront.net/data/2021/12/06/【OS 直営】app_自転車画像-_HC.png
バッテリー残量（正規化値）	current_fuel_percent	○	Number	正規化されたバッテリー残量値	例：50
推定走行可能距離	current_range_meters	○	Number	現在のバッテリー残量などから推定される走行可能距離	例：0
最終利用日時	last_used	○	Datetime	当該自転車が最後に利用された日時	例：null

➢ レスポンスサンプル


```
{
  "data": {
    "id": "6113",
    "name": [
      {
        "language": "JA",
        "text": "竜泉 1 丁目"
      }
    ],
    "brand_name": "HELLO CYCLING",
    "name_company": [
      {
        "language": "JA",
        "text": "OpenStreet 株式会社"
      }
    ],
    "url": "https://resource.hellocycling.jp/webview/company/detail/11",
    "system_id": "11",
    "brand_image_url":
"https://d2a2mitt2l0zmn.cloudfront.net/data/2019/05/07/keep_aspect_ratio/120/HC-logo.png",
    "address": "東京都台東区竜泉 1-33-7",
    "opening_hour": "00:00",
    "station_opening_hours": "24h",
    "has_parking_limit": true,
    "num_docks_available": 6,
    "num_vehicles_available": 3,
    "num_vehicles_reserved": 0,
    "lat": 35.724652,
    "lon": 139.789753,
    "is_charging_station": true,
    "post_code": "110-0012",
    "vehicles": [
      {
        "id": "20104",
        "vehicle_type_id": "2",
        "price": 160,
```

```
            "hour_price": 0,
            "day_price": null,
            "half_day_price": 2500,
            "photo_path": "https://d2a2mitt2l0zmn.cloudfront.net/data/2021/12/06/【OS 直営】app_自転車画像-_HC.png",
            "current_fuel_percent": 50,
            "current_range_meters": 0,
            "last_used": null
        }
    ]
  }
}
```


【IF005】ポート検索標準 API＜既存改修＞

- 概要
  - シェアサイクルアプリから送信される認証トークンを受信し、ポート ID、緯度・経度、貸出可能自転車台数などの情報を一覧で送信する
- 本インターフェースを利用する機能
  - 【FN007】ポート検索
- 共通仕様
  - プロトコル
    - HTTPS
  - メソッド
    - GET
  - リクエスト形式
    - クエリ
  - レスポンス形式
    - JSON
  - 文字コード
    - UTF-8
  - リクエストクエリ

項目	名称	必須	データ型	説明	値
認証トークン	rest_token	-	String	ポート検索時の認証のために発行されるトークン	例：7c74bcde4f03db34af7acbbe7f6b321d

- レスポンスパラメータ

項目	名称	必須	データ型	説明	値
レスポンスコード	response	○	Integer	処理結果を示すステータスコード	1：成功、0：失敗(エラー)、500：サーバーエラー
レスポンスデータ	data	○	Array	レスポンスデータ	-
ポート ID	id	○	Integer	ポートを一意に識別するための ID	例：17
駐輪可能台数	num_docks_available	○	Integer	当該ポートに駐輪可能な自転車の台数	例：5
貸出可能台数	num_vehicles_available	○	Integer	当該ポートにおいて現在貸出可能な状態にある自転車の台数	例：3
予約済み自転車台数	num_vehicles_reserved	-	Integer	当該ポートにおいてすでに予約されている自転車の台数	例：0
更新時刻	update_time	○	Integer	情報が更新された最新の時刻	例：1753344971

- レスポンスサンプル

```
{
  "response": 1,
  "data": [
    {
      "id": 17,
      "num_docks_available": 5,
      "num_vehicles_available": 3,
      "num_vehicles_reserved": 0,
      "update_time": 1753344971
    }
  ]
}
```

【IF006】予約登録標準 API＜既存改修＞

- 概要
  - 認証トークン及び自転車 ID を受信し、自転車の予約登録を行い、予約登録結果を返信する
  - 予約登録に伴って貸出可能な自転車台数が減少するため、ポート情報リアルタイム連携機能により、減少後の貸出可能自転車台数をメッセージとして送信する
- 本インターフェースを利用する機能
  - 【FN002】予約・貸出・返却
- 【FN003】ポート情報リアルタイム連携共通仕様
  - プロトコル
    - HTTPS
  - メソッド
    - POST
  - リクエスト形式
    - JSON
  - レスポンス形式
    - JSON
  - 文字コード
    - UTF-8
  - リクエストパラメータ

項目	名称	必須	データ型	説明	値
自転車 ID 一覧	vehicle_ids	○	Array	自転車 ID の配列	-
自転車 ID	vehicle_id	-	string	自転車 ID	例：20105
ポート ID	station_id	○	string	ポートを一意に識別するための ID	例：6113
認証トークン	rest_token	○	String	予約操作時の認証のために発行されるトークン	例：7c74bcde4f03db34af7acbbe7f6b321d

- レスポンスパラメータ

項目	名称	必須	データ型	説明	値
レスポンス	response	○	Integer	処理結果を示すステータスコード	1：成功、0：失敗(エラー)、500：サーバーエラー
レスポンスデータ	data	○	Object	レスポンスデータ	-
ポート情報	station	○	Object	ポート情報を格納するオブジェクト	-
ポート ID	station_id	○	String	ポートを一意に識別するための ID	6113
ポート名	name	○	Array	利用者に表示されるポートの名称	-
言語コード	language	○	String	言語コード	JA
ポート名	text	○	String	ポート名	竜泉 1 丁目
注文情報一覧	orders	○	Array	注文情報の一覧	-
注文 ID	id	○	Integer	注文を一意に識別するための ID	例：87263039
自転車 ID	vehicle_id	○	Stringr	自転車を一意に識別するための ID	例：20105

- レスポンスサンプル


```
{
  "response": 1,
  "data": {
    "station": {
      "station_id": "6113",
      "name": [
        {
          "language": "JA",
          "text": "竜泉 1 丁目"
        }
      ]
    },
    "orders": [
      {
        "id": 87263039,
        "vehicle_id": "20105"
      }
    ]
  }
}
```


【IF007】予約キャンセル標準 API＜既存改修＞

- 概要
  - 認証トークン及び予約 ID を受信し、自転車の予約登録を取り消し、予約取り消し結果を返信する
  - 予約取り消しに伴って貸出可能な自転車台数が増加するため、ポート情報リアルタイム連携機能により、増加後の貸出可能自転車台数をメッセージとして送信する
- 本インターフェースを利用する機能
  - 【FN002】予約・貸出・返却
- 【FN003】ポート情報リアルタイム連携共通仕様
  - プロトコル
    - HTTPS
  - メソッド
    - POST
  - リクエスト形式
    - JSON
  - レスポンス形式
    - JSON

- 文字コード
  - UTF-8
- リクエストパラメータ

項目	名称	必須	データ型	説明	値
注文 ID	reservation_id	○	Integer	予約処理を一意に識別するための ID	例：87263039
認証トークン	rest_token	○	String	予約操作時の認証のために発行されるトークン	例：7c74bcde4f03db34af7acbbe7f6b321d

- レスポンスパラメータ

項目	名称	必須	データ型	説明	値
レスポンス	response	○	Integer	処理結果を示すステータスコード	1：成功、0：失敗(エラー)、500：サーバーエラー

- レスポンスサンプル

```
{
 "response": 1,}
```

【IF008】解錠処理標準 API＜既存改修＞

- 概要
  - 認証トークン及び注文 ID、スマートロックの MAC アドレスを受信し、スマートロックを解錠し、解錠結果を返信する
- 本インターフェースを利用する機能
  - 【FN002】予約・貸出・返却
- 【FN003】ポート情報リアルタイム連携共通仕様
  - プロトコル
    - HTTPS
  - メソッド
    - POST
  - リクエスト形式
    - JSON
  - レスポンス形式
    - JSON
  - 文字コード
    - UTF-8

- リクエストパラメータ

項目	名称	必須	データ型	説明	値
認証トークン	rest_token	○	String	解錠処理時の認証のために発行されるトークン	例：7c74bcde4f03db3 4af7acbbe7f6b321d
端末 MAC アドレス	mac_address	○	String	シェアサイクルアプリを利用する端末の MAC アドレス	例：8C:59:DC:FC:EC:A0
注文 ID	order_id	○	Integer	注文を一意に識別するための ID	例：87263162

- レスポンスパラメータ

項目	名称	必須	データ型	説明	値
レスポンス	response	○	Integer	処理結果を示すステータスコード	1：成功、0：失敗(エラー)、500：サーバーエラー
メッセージ	msg	○	String	処理結果やエラー内容などを補足的に説明するためのメッセージ	NOT_ON_RESERVE_OR_RENTAL：予約済みもしくはレンタル中ではないため、ロック解除できません、MAC_ADDRESS_NOT_MATCH：MAC アドレス突き合い不一致エラー、RENTAL_FAILED：レンタル失敗、UNLOCK_FAILED：解錠失敗、ALREADY_UNLOCK：すでに解錠しています、OLD_VERSION：白いロックの解錠は PIN コードを入力して解錠してください、NOT_NEAR：お客様が自転車の近くにいないようです。自転車のお近くで再度お試しください DEVICE_NOT_CONNECT：スマートロックとの接続が失敗しました

- レスポンスサンプル

```
 {
 "response": 1,
 "msg": "解錠中"
}
```

【IF009】返却処理標準 API＜既存改修＞

- 概要
  - 認証トークン及び注文 ID を受信し、モビリティの返却を記録し、返却結果を送信する
  - 返却処理に伴って貸出・返却可能な自転車台数が変化するため、ポート情報リアルタイム連携機能により、変化後の貸出・返却可能台数をメッセージとして送信する
- 本インターフェースを利用する機能
  - 【FN002】予約・貸出・返却
- 【FN003】ポート情報リアルタイム連携共通仕様
  - プロトコル
    - HTTPS
  - メソッド
    - POST
  - リクエスト形式
    - JSON
  - レスポンス形式
    - JSON
  - 文字コード
    - UTF-8
  - リクエストパラメータ

項目	名称	必須	データ型	説明	値
注文ID	order_id	○	Integer	注文を一意に識別するための ID	例：87263162
認証トークン	rest_token	○	String	注文操作時の認証のために発行されるトークン	例：7c74bcde4f03db34af7acbbe7f6b321d

- レスポンスパラメータ

項目	名称	必須	データ型	説明	値
レスポンス	response	○	Integer	処理結果を示すステータスコード	1：成功、0：失敗(エラー)、500：サーバーエラー
メッセージ	msg	-	String	処理結果やエラー内容などを補足的に説明するためのメッセージ	NO_ORDER：該当注文は存在しません、 UNAUTHORIZATION：認可エラー、 NOT_ON_RENTA：レンタル中ではないため、返却できません、 LOST_STATUS：施錠が確認できないため返却ができません。自転車に鍵をかけた状態で返却をお試しください、 UNLOCK：施錠してから、再度返却ボタンを押してください、 LOST_LOCATION：位置情報を取得できません、 OUT_OF_STATION：自転車がステーション外にあるため返却できません、 NO_VACANCY：現在ステーションが満車のため返却ができません。空きがある別のステーションへ自転車を移動させて返却をお試しください、 OUT_OF_SERVICE：現在ステーションが営業時間外のため返却ができません。返却可能な別のステーションへ自転車を移動させて返却をお試しください、 MISMATCH_PORT_CATEGORY：現在ご利用中の車両はこちらのステーションには返却できません。詳しくはメッセージをご確認ください、 ERROR：返却失敗

- レスポンスサンプル

```
{
 "response": 1,
 "msg": "返却完了"
}
```

2-5. ユーザーインターフェース（UI）

2-5-1. 画面遷移図

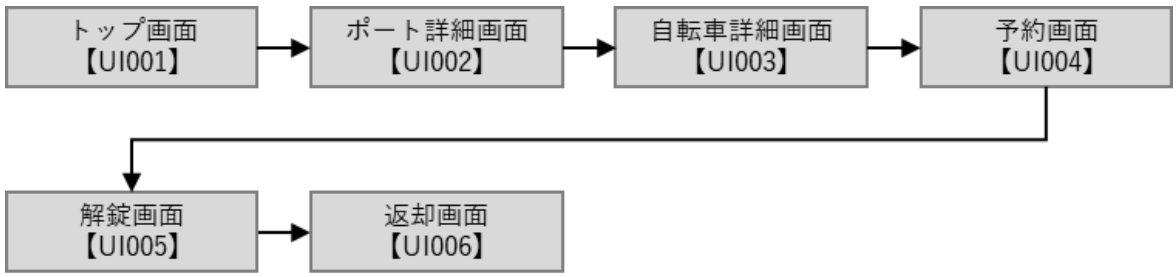


図 2-21 【HW002】スマートフォン用画面遷移図

2-5-2. ユーザーインターフェース一覧

表 2-6 ユーザーインターフェース一覧

※朱文字：新規開発・既存改修

ID	画面名	説明	画面を表示した機能（ID）
UI001	トップ画面	● シェアサイクルアプリ起動時に表示する	FN006 FN007
UI002	ポート情報画面	● 選択したポートの位置情報、貸出・返却可能台数、稼働時間、対応モビリティ種別などの詳細情報を表示する。地図上にも位置を表示する	FN003 FN006
UI003	自転車詳細画面	● 選択した車体のバッテリー残量、現在地、状態（貸出中／空き）などをテキスト形式で表示する	FN006
UI004	予約画面	● ポートとモビリティを選択して予約を行う画面。予約可能な台数や制限事項などを表示し、予約の完了・キャンセルを操作ができる	FN002 FN006 FN007
UI005	解錠画面	● 予約完了後に車体を解錠する操作を提供する画面。スマートロックと連携し、指定モビリティを解錠できる	FN002 FN003
UI006	返却画面	● 利用終了時に車体を返却する画面。返却先のポートが正しく選ばれているかを確認すると同時に空き状況や返却可否を確認し、返却操作を行う	FN002 FN003

2-5-3. ユーザーインターフェースの詳細

ユーザーインターフェース（画面）の詳細を記す。なお、本業務において開発（新規・改修）を行うユーザーインターフェース（画面）を朱文字で示す。

【UI001】トップ画面＜既存改修＞

図 2-22 Hello Cycling トップ画面

- 概要
  - ポートの一覧表示や検索を行う画面
  - 地図ベースで利用可能なポートを表示し、ポートを選択することでポートや自転車の詳細情報を確認可能
  - 自社ポートだけでなく、他事業者の共同利用ポートも地図上に一覧表示・検索する
  - また、他事業者による利用や予約・返却操作が発生した場合、MQTT 通知を通じて空きドック数や貸出可能台数をリアルタイムで反映する
- 本画面から利用する機能
  - 【FN006】ポート基本情報取得
  - 【FN007】ポート検索

【UI002】ポート情報画面＜既存改修＞

図 2-23 Hello Cycling ポート情報画面

- 概要
  - トップ画面で選択されたポートの詳細情報（位置情報、貸出・返却可能台数、稼働時間、対応モビリティ種別など）を表示する画面
  - 自社ポートだけでなく、他事業者の共同利用ポートの詳細を表示する
  - 共同利用ポートの場合は、他事業者から MQTT 通知によりリアルタイムで受信したステータス情報が表示される
- 本画面から利用する機能
  - 【FN003】ポート情報リアルタイム連携
  - 【FN006】ポート基本情報取得

【UI003】自転車詳細画面

図 2-24 Hello Cycling 自転車詳細画面

- 概要
  - 選択された自転車の詳細情報（バッテリー残量や車体種別など）を表示する画面
- 本画面から利用する機能
  - 【FN006】ポート基本情報取得

【UI004】予約画面

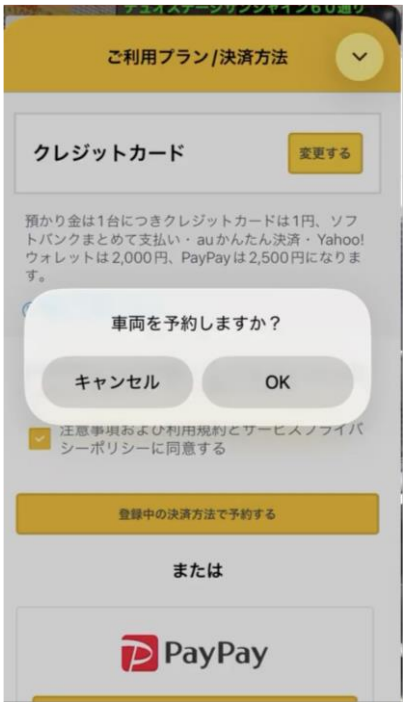


図 2-25 Hello Cycling 予約画面

- 概要
  - 自転車を事前に予約する画面
- 本画面から利用する機能
  - 【FN002】予約・貸出・返却
  - 【FN006】ポート基本情報取得
  - 【FN007】ポート検索

【UI005】解錠画面

図 2-26 Hello Cycling 解錠画面

- 概要
  - 予約済み又は即時利用の自転車に対して、スマートロックを操作して解錠を行う画面
  - 予約確定後、リアルタイムでポートの空き状況が更新され、他事業者にも通知される
- 本画面から利用する機能
  - 【FN002】予約・貸出・返却
  - 【FN003】ポート情報リアルタイム連携

【UI006】返却画面＜既存改修＞

図 2-27 Hello Cycling 返却

- 概要
  - 利用者が自転車の利用を終了し、返却操作を行うための画面
  - 自社ポートだけでなく、他事業者の共同利用ポートにも返却できる
  - 利用終了時に返却操作を行うと、ポートの空き状況がリアルタイムで更新され、MQTT 通知により他事業者にも反映される
- 本画面から利用する機能
  - 【FN002】予約・貸出・返却
  - 【FN003】ポート情報リアルタイム連携

3. シェアサイクルポート共有 API システム：非機能要件（NF）

3-1. 非機能要件一覧

表 3-1 非機能要件一覧

カテゴリ	ID	非機能項目	要件詳細
拡張性	NFR001	スケーラビリティと拡張性	● 将来的に生じるデータ量の増加や新機能の追加に対応できるよう、Amazon EC2 のオートスケーリング機能を用いる
継続性	NFR002	データのバックアップ	● データベースは毎日バックアップを取得する
応答性能	NFR003	レスポンスタイム	● レスポンスタイムは 3 秒以内とする
信頼性	NFR004	稼働率	● 月間の稼働率は 98%以上とする

3-2. 非機能要件の詳細

【NFR001】スケーラビリティと拡張性

- 概要
  - 将来的に生じるデータ量の増加や新機能の追加に対応できるよう、Amazon EC2 のオートスケーリング機能を用いる
- 設定理由
  - ポート共有システムは将来のデータ量の増加や新機能の追加に対応する必要があるため、負荷が増加した際にも柔軟に対応できるよう、スケーラビリティと拡張性の高い構成が必要である

【NFR002】データのバックアップ

- 概要
  - データベースは毎日バックアップを取得する
- 設定理由
  - 障害発生時の早期復旧や、連携データを喪失して事業の継続が困難になるといったリスクを回避するため

【NFR003】レスポンスタイム

- 概要
  - レスポンスタイムは 3 秒以内とする
- 設定理由
  - 利用者がストレスなく操作できる UX を提供できる秒数を想定する

【NFR004】稼働率

- 概要
  - 月間の稼働率は 97.7%以上とする
- 設定理由
  - 常にアクセスが可能な状態を維持し、安定したサービスの提供を行う必要があるため

4. 実証調査に利用するデータ（DT）

4-1. 実証調査に利用するデータ一覧

表 4-1 実証調査に利用するデータ一覧

※朱文字：本実証で変換・作成するデータ

ID	データ名称	データ形式	出所	データを利用する ID
DT001	OS 社ポート情報	JSON	Openstreet 株式会社	FN001
DT002	Lime 社ポート情報	JSON	Lime 株式会社	FN001
DT003	共有ポート情報	JSON	シェアサイクル事業者	FN001、FN002、FN003、FN005、FN006
DT004	ポートステータス	JSON	シェアサイクル事業者	FN001、FN00、FN003

4-2. 実証調査に利用するデータの詳細

実証調査に利用するデータの詳細を記す。なお、本業務において変換・生成を行うデータを朱文字で示す。

【DT001】OS 社ポート情報

- 概要
  - Openstreet が保有するポートの情報
- データ定義

表 4-2 Openstreet 社ポート情報のサンプル

No.	日本語名称	フィールド名	データ型	書式・選択肢など
1	ポート ID	station_id	string	例：“9701”
2	ポート名（和名）	text	string	例：“〇〇駅自転車駐車場”
3	ポート名（英名）	language	string	例：“〇〇 Station bicycle parking lot”
4	位置情報（緯度）	lat	number	例：XX.XXXXXX
5	位置情報（経度）	lon	number	例：XXX.XXXXXX
6	場所（住所）	address	string	例：“神奈川県横浜市〇〇”
7	営業開始時間	opening_hour	string	例：“0:00”
8	最大駐輪可能台数	has_parking_limit	non-negative integer	例：5
9	共有開始日時	share_start_at	string	例：“YYYY-MM-DDThh:mm:ssZ”
10	共有解除日時	share_end_at	string	例：“YYYY-MM-DDThh:mm:ssZ” 解除しない場合は記載不要
11	運用状態	status_before_sharing	string	例：“運用中”
12	営業終了時間	station_closing_time	string	例：“0:00”
13	外観画像 URL	image_url	URL	例：“https://~.jpg”
14	パートナーポート ID	partner_station_id	ID	例：500
15	ステーション表示	is_displayed	boolean	例：true
16	DBS ビーコン	beacons	array<string>	-

	ID1			
17	DBS ビーコン ID2	beacons	array<string>	-
18	DBS ビーコン ID3	beacons	array<string>	-
19	DBS ビーコン ID4	beacons	array<string>	-
20	DBS ビーコン ID5	beacons	array<string>	-
21	DBS ビーコン ID6	beacons	array<string>	-
22	DBS ビーコン ID7	beacons	array<string>	-
23	DBS ビーコン ID8	beacons	array<string>	-
24	DBS ビーコン ID9	beacons	array<string>	-
25	DBS ビーコン ID10	beacons	array<string>	-
26	定休日（曜日別）	regular_holiday	array<string>	例：“無”
27	臨時休業日	temporary_closures	array<string>	例：“無”
28	POP タイトル	pop_title	string	-
29	POP 本文	pop_body	string	-
30	POP 画像 URL	pop_image_url	URL	-
31	OS ビーコン 1	beacons	array<string>	-
32	OS ビーコン 2	beacons	array<string>	-

- データ形式
  - JSON
- 出所
  - Openstreet 株式会社

【DT002】Lime 社ポート情報

- 概要
  - Lime 社が保有するポートの情報
- データ定義

表 4-3 Lime 社ポート情報のサンプル

No.	日本語名称	フィールド名	データ型	書式・選択肢など
1	座標（経緯度）	geometry.coordinates	array of numbers	例：139.707905, 35.737876
2	ID	id	object	例：URTYKPI2EIQSW
3	収容台数	capacity	integer	例：2

- データ形式
  - JSON
- 出所
  - Lime 株式会社

【DT003】共有ポート情報

- 概要
  - 複数事業者で共同利用するポートの情報
- データ定義

表 4-4 共有ポート情報のサンプルイメージ

No.	日本語名称	フィールド名	データ型	書式・選択肢など
1	共有開始日時	share_start_at	string	例：“1745408031”
2	共有解除日時	share_end_at	string	例：“1756652400”
3	ポート ID	station_id	number	例：“00000001”
4	共用企業ポート ID	provider_station_id	number	例：“999999”
5	ポート名（和名）	name_ja	string	例：“A ステーション”
6	ポート名（英名）	name_en	string	例：“A station”
7	運用状態	status	string	例：“運用中”, ”停止中”
8	公開・非公開	is_visible	boolean	例：true
9	場所（住所）	address	string	例：“神奈川県横浜市〇〇”
10	位置情報（緯度）	lat	number	例：XX.XXXXXX
11	位置情報（経度）	lon	number	例：XXX.XXXXXX
12	最大駐輪可能台数（設置許容数）	capacity	number	例：5
13	ポート外観画像 URL	image_url	string	例："https://~.jpg"
14	営業開始時間	opening_hour	string	例：“0:00”
15	営業終了時間	close_time	string	例：“0:00”
16	注意事項	note	string	例：“公園内は走行禁止”

- データ形式
  - JSON
- 出所
  - シェアサイクル事業者

【DT004】ポートステータス

- 概要
  - 貸出・返却可能な自転車台数
- データ定義

表 4-5 ポートステータスのサンプルイメージ

No.	日本語名称	フィールド名	データ型	書式・選択肢など
1	事業者 ID	system_id	ID	例：“OS”
2	ポートステータス	station_details	array<string>	-
3	ポート ID	stationid	string	例：997857
4	ステータス更新のタイムスタンプ	update_time	timestamp	例：1745408031
5	現在の貸出可能台数	um_vehicles_available	integer	例：3
6	現在の予約数	num_vehicles_reserved	integer	例：2
7	バッテリー残量が少ない台数	low_battery_num	integer	例：1

- データ形式
  - JSON
- 出所
  - シェアサイクル事業者

5. 用語集

用語	定義・説明
共同利用ポート	● あるシェアサイクル事業者のシェアサイクルポートを他のシェアサイクル事業者と共同利用するポート
共有ポート情報	● ポート情報のうち、共同利用ポートの情報
シェアサイクルポート	● シェアサイクルの貸出・返却が可能な駐輪場
GBFS	● General Bikeshare Feed Specification. マイクロモビリティの標準データフォーマット
ポート情報	● シェアサイクルポートの ID、名称、位置、貸出可能台数、駐輪可能台数などの情報
ポートステータス	● ポート情報のうち、貸出可能台数、駐輪可能台数などの動的な情報

COMmmONS by MLIT

シェアサイクルポート共有API システム設計書
Ver1.0

発行日:　2026年3月
委託者:　国土交通省 総合政策局
　　モビリティサービス推進課
受託者:　パシフィックコンサルタンツ株式会社
　　OpenStreet株式会社