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　ヘルスケアMaaS実証プロジェクト
ヘルスケアMaaSシステム 技術検証レポート付録

ヘルスケアMaaSシステム
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.003-01

目次

1. 開発スコープ

1-1. 概要

病院の予約システムとデマンドバスの配車システムを連携させることで通院時（特に、往路）の交通手段を確保するため、病院の診療終了時刻を予測する「離院時刻予測モデル」とデマンドバス配車システムの連携インターフェースとなる「ヘルスケア MaaS Gateway」を開発する。離院時刻予測モデルは、病院の診療が終わった時刻（離院時刻）にデマンドバスの配車を実現するため、電子カルテシステムから取得した実績データ等を基に離院時刻を予測するモデルであり、Personal Health Record（以下、「PHR」という。）システム上に構築する。また、離院時刻を予測するため、電子カルテシステムから抽出した各診療イベントの進捗実績データ（来院時刻、診察開始時刻、診察終了時刻、検査実施時間、会計完了時刻などを記録した電子カルテシステム上のデータ）を収集し、診察当日の診療イベントに応じた在院時間の傾向を分析する。

離院時刻の予測精度向上のためには、通常の診察予約に加え、診察前、診察後の各イベントに掛かる時間を含めて予測する必要があるため、診察時間以外の観点を追加し以下のデータを収集及び分析を行う。

- 診療科・曜日・診察予約時間ごとの患者数や診察時間を収集し、診察所要時間の変動を分析
- 各検査の平均所要時間を収集し、検査内容による在院時間を分析
- 各処置の平均所要時間を収集し、処置内容による在院時間を分析
- 会計待ち時間を曜日、時間帯ごとに収集し、会計に掛かる時間の変動を分析
- 上記のデータ収集、分析により、在院時間の算定における重要な特徴量を選定し各特徴量と在院時間の相関をルール化する

上記で策定したルールに沿って、診察当日の特徴量に応じた在院時間を算定し、離院時刻を予測する。

ヘルスケア MaaS Gateway は、病院予約システムとデマンドバス配車システムを連携させ、病院の診察予約時刻に応じてデマンドバスの往路配車予約及び離院時刻予測モデルで出力された離院予測時刻を用いてデマンドバスの復路配車予約を行う。また、利用者の希望に応じて、病院の離院後、薬局やスーパーなど帰宅するまでに立ち寄る場所を経由した配車予約も可能とする。

それぞれの配車予約を行うに際し、ヘルスケア MaaS Gateway で予約情報を生成する。病院の往路予約では、診察予約時刻をデマンドバスの病院到着希望時刻として予約情報を生成し、復路予約では離院予測時刻をベースにデマンドバスの病院出発希望時刻として予約情報を生成する。

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

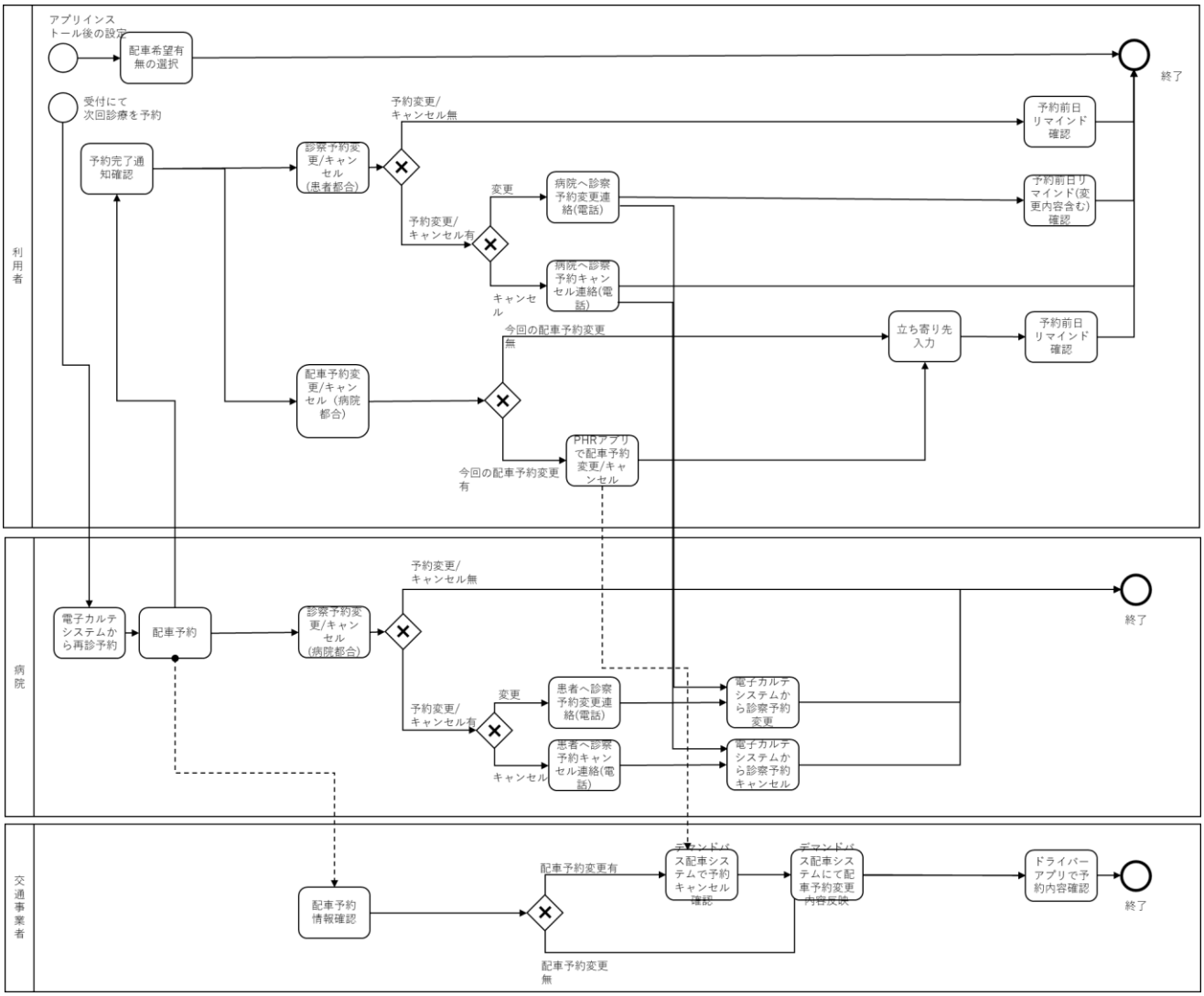


図 1-1 業務フロー（配車希望選択・次回診療・配車予約～乗車前日）

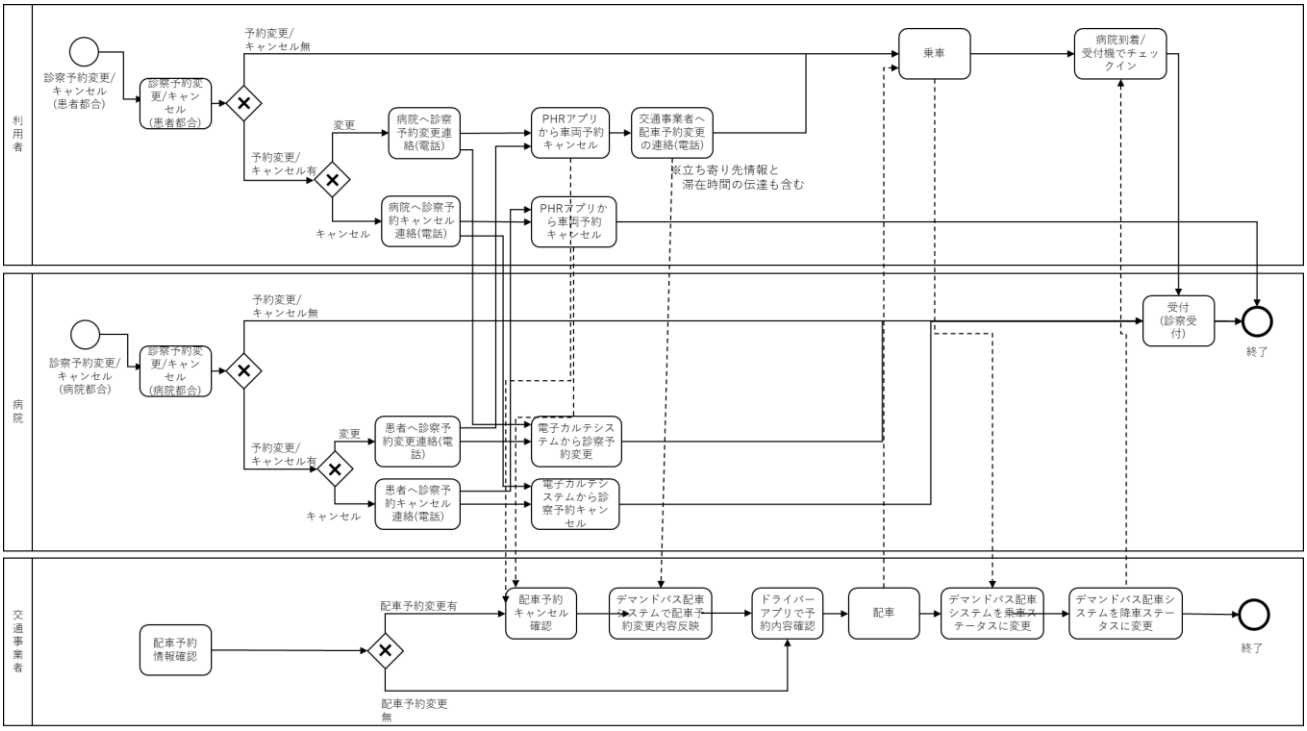


図 1-2 業務フロー（診察当日：乗車～病院到着）

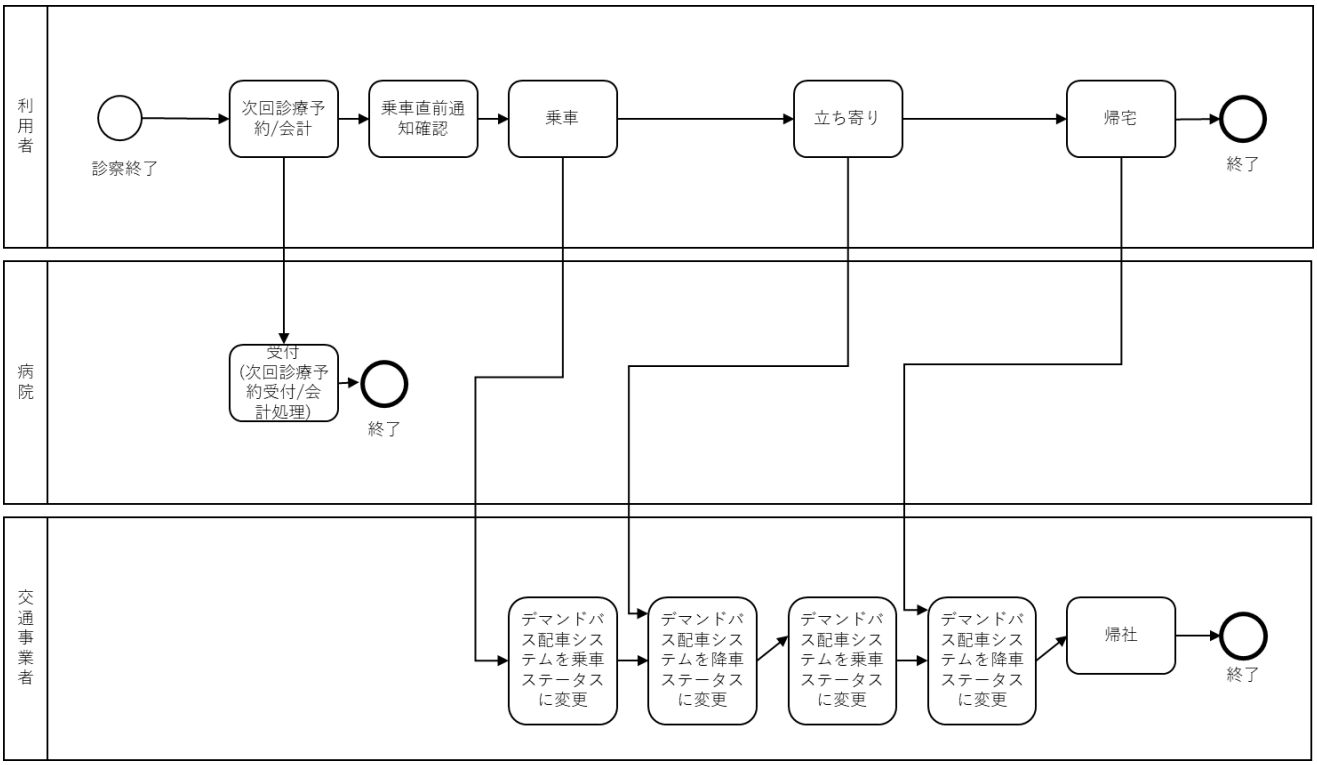
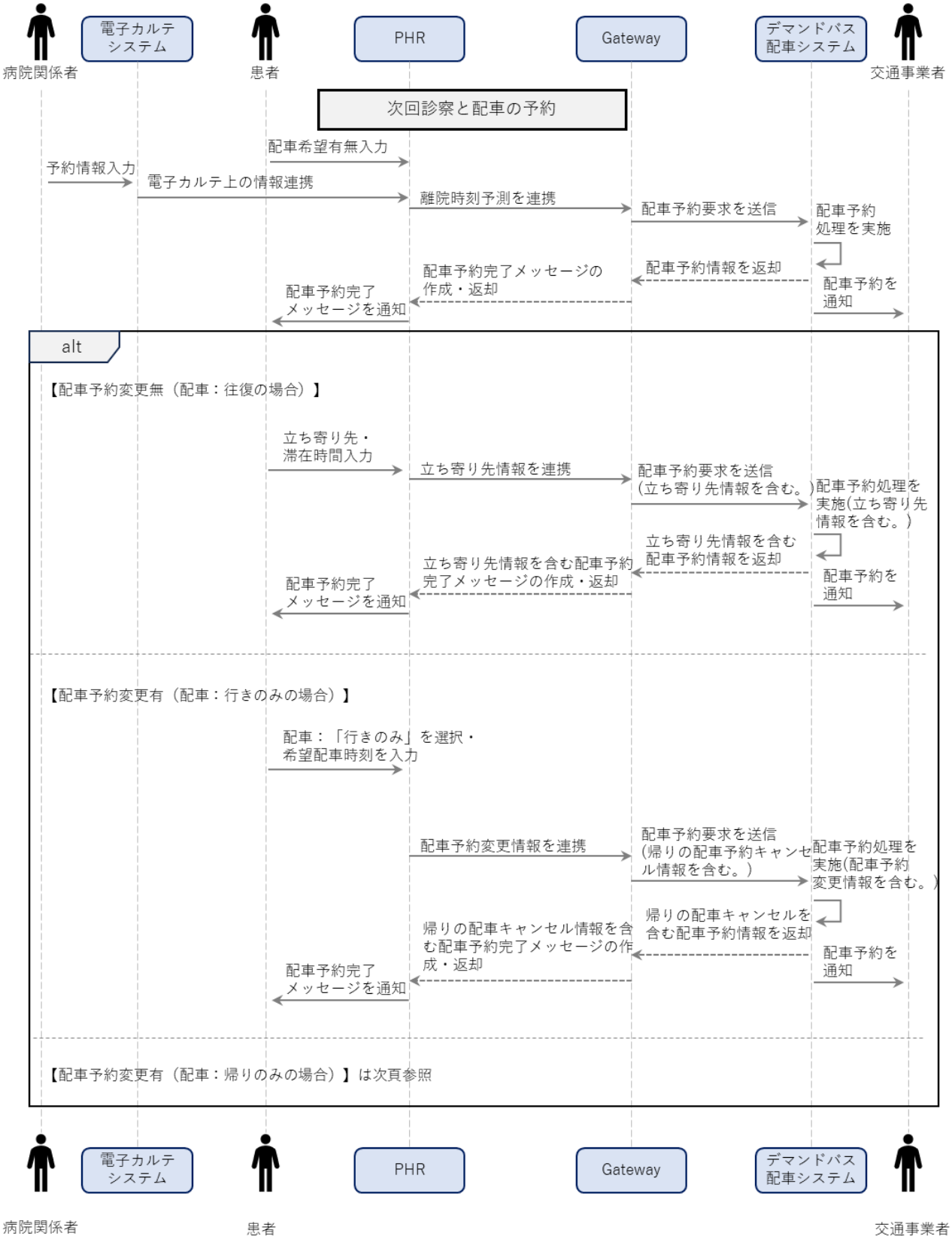


図 1-3 業務フロー（診察終了～帰宅済）

2．システムシーケンス図

① 次回診察と配車予約時の処理



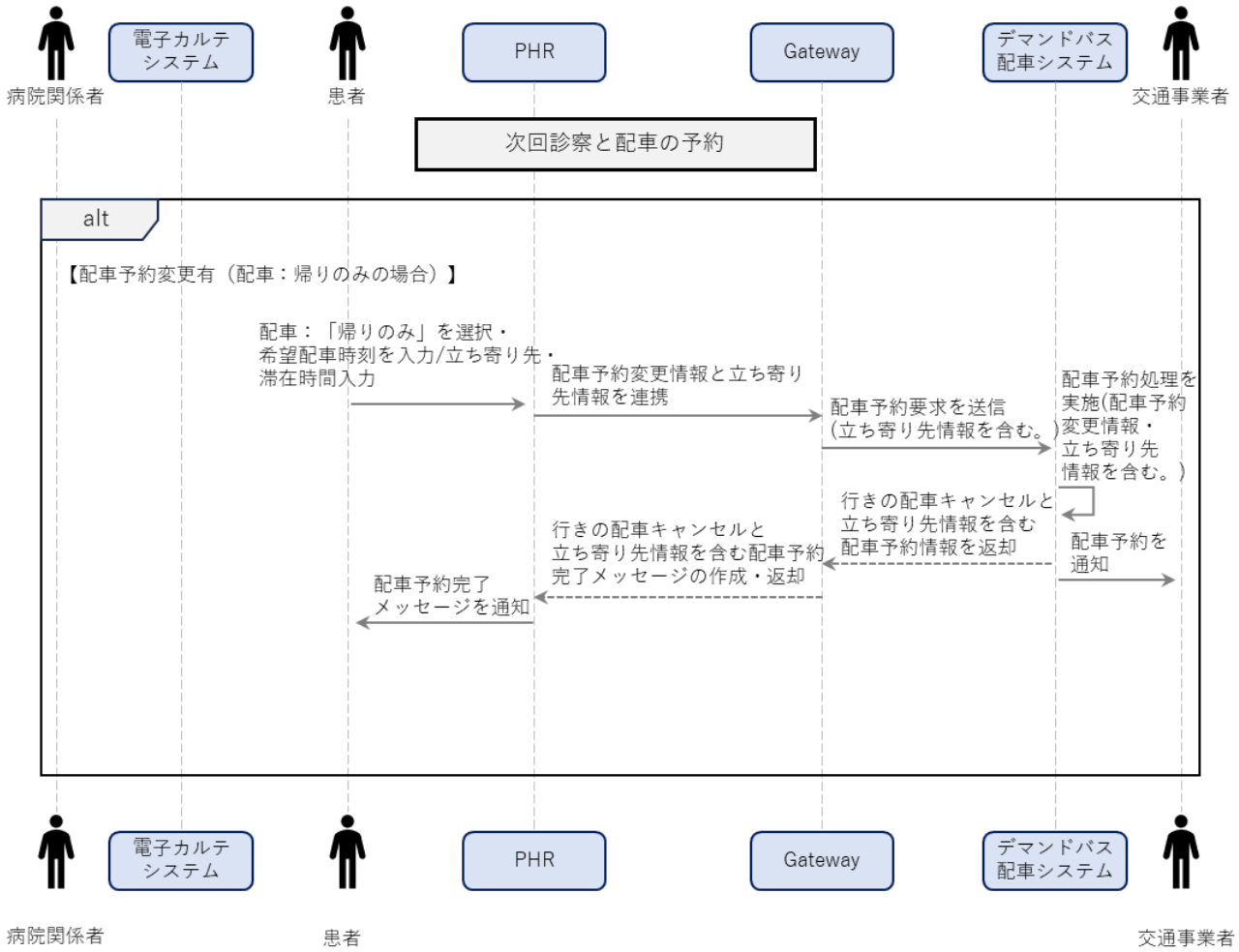
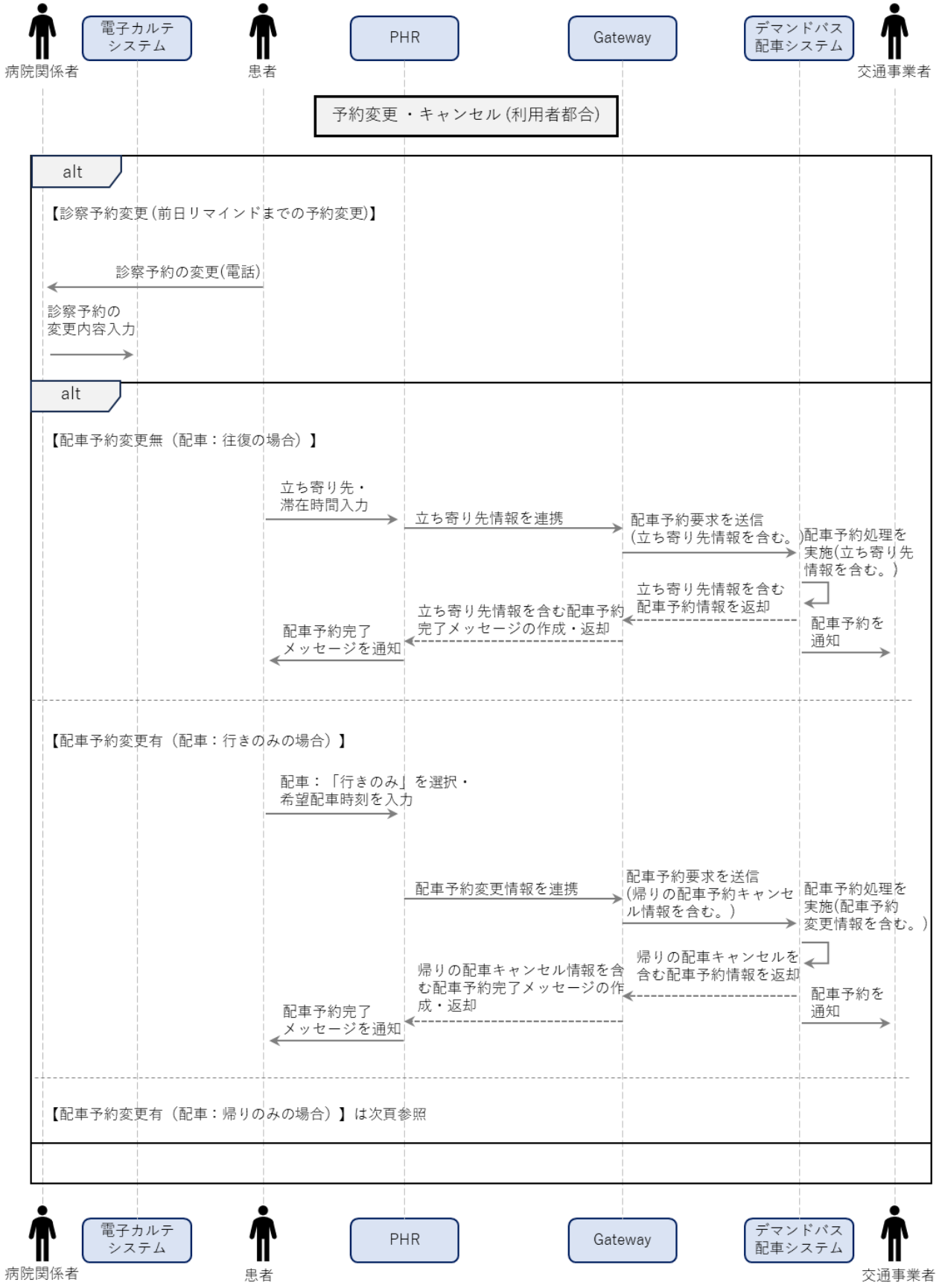


図 1-4 次回診察と配車予約時の処理

② 予約変更・キャンセル時(利用者都合)の処理



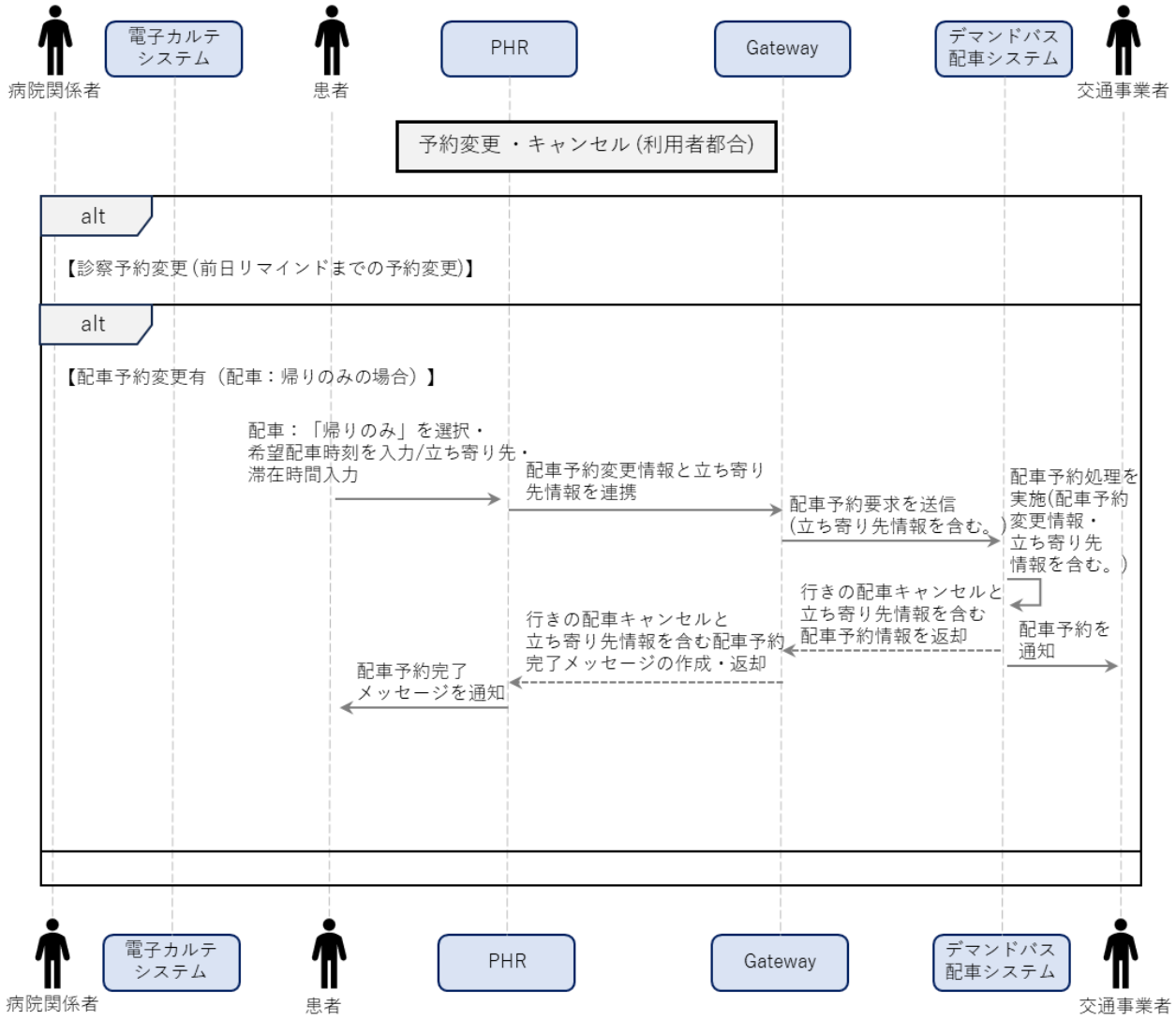
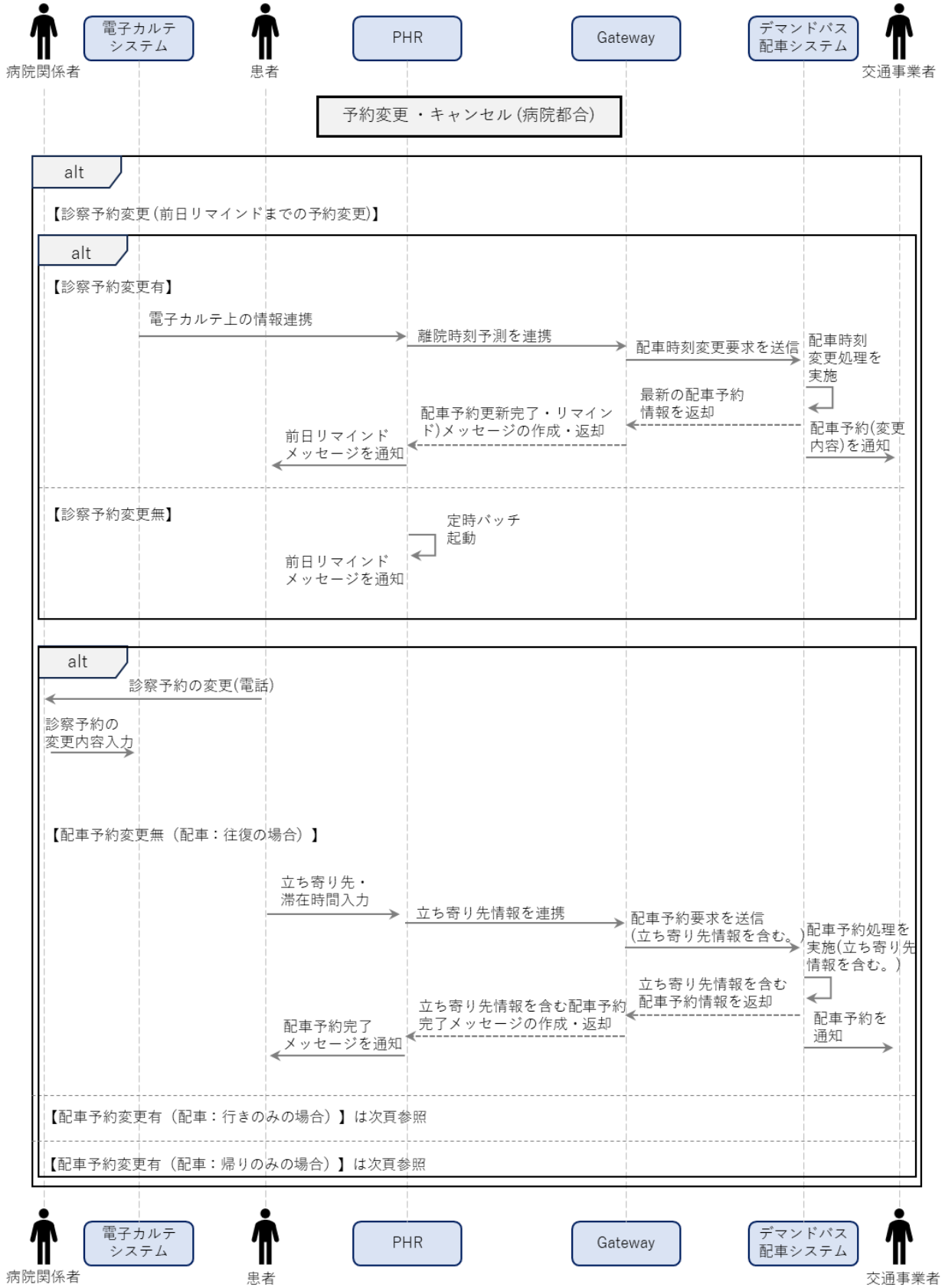
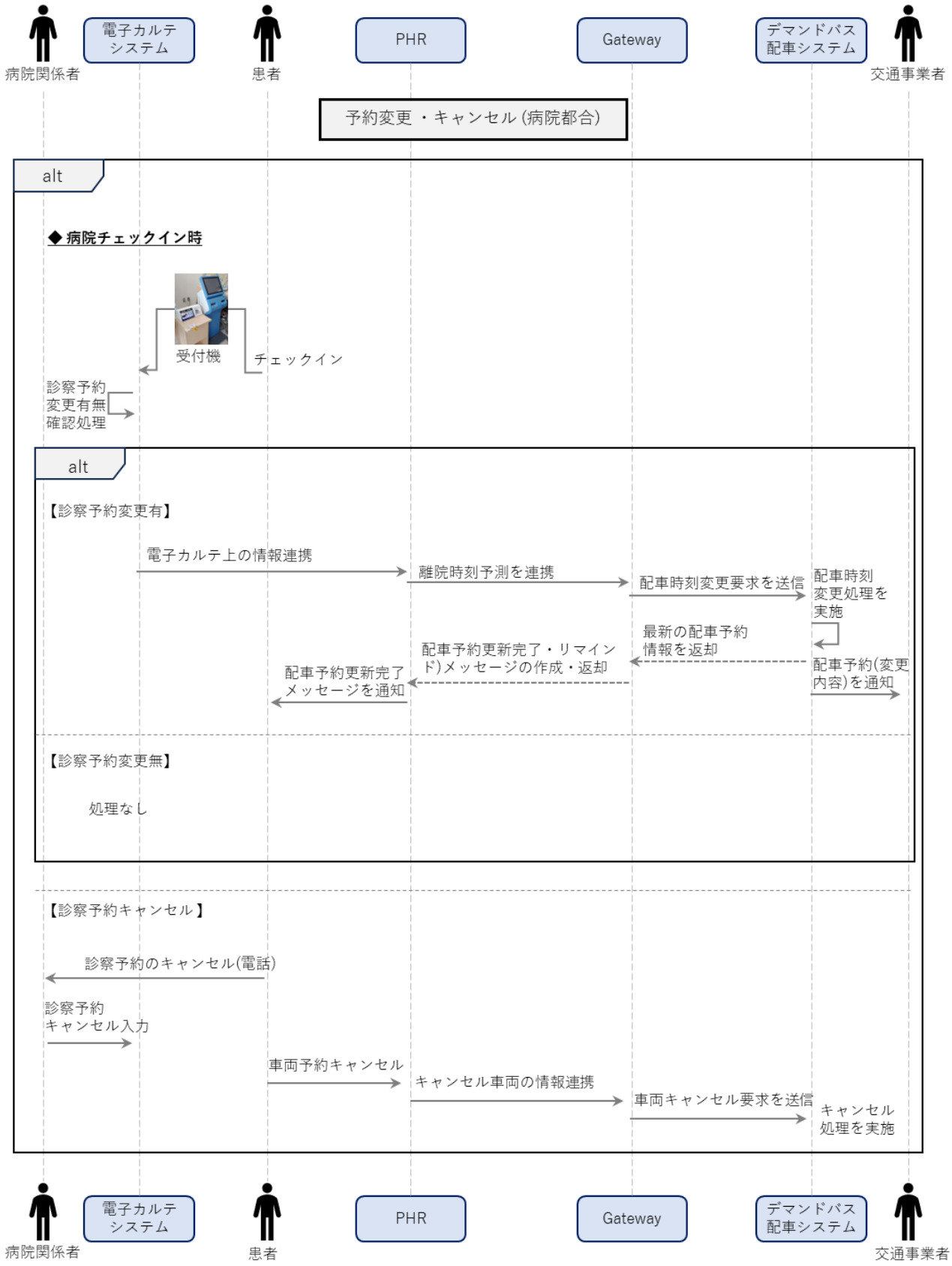


図 1-5 予約変更・キャンセル時(利用者都合)の処理

③ 予約変更・キャンセル時(病院都合)の処理




病院関係者
電子カルテ
システム
患者
PHR
Gateway
デマンドバス
配車システム
交通事業者
予約変更・キャンセル(病院都合)
alt
◆病院チェックイン時
受付機
チェックイン
診察予約
変更有無
確認処理
alt
【診察予約変更有】
電子カルテ上の情報連携
離院時刻予測を連携
配車時刻変更要求を送信
配車時刻
変更処理を
実施
最新の配車予約
情報を返却
配車予約更新完了・リマイン
ド)メッセージの作成・返却
配車予約更新完了
メッセージを通知
配車予約(変更
内容)を通知
【診察予約変更無】
処理なし
【診察予約キャンセル】
診察予約のキャンセル(電話)
診察予約
キャンセル入力
車両予約キャンセル
キャンセル車両の情報連携
車両キャンセル要求を送信
キャンセル
処理を実施
病院関係者
電子カルテ
システム
患者
PHR
Gateway
デマンドバス
配車システム
交通事業者

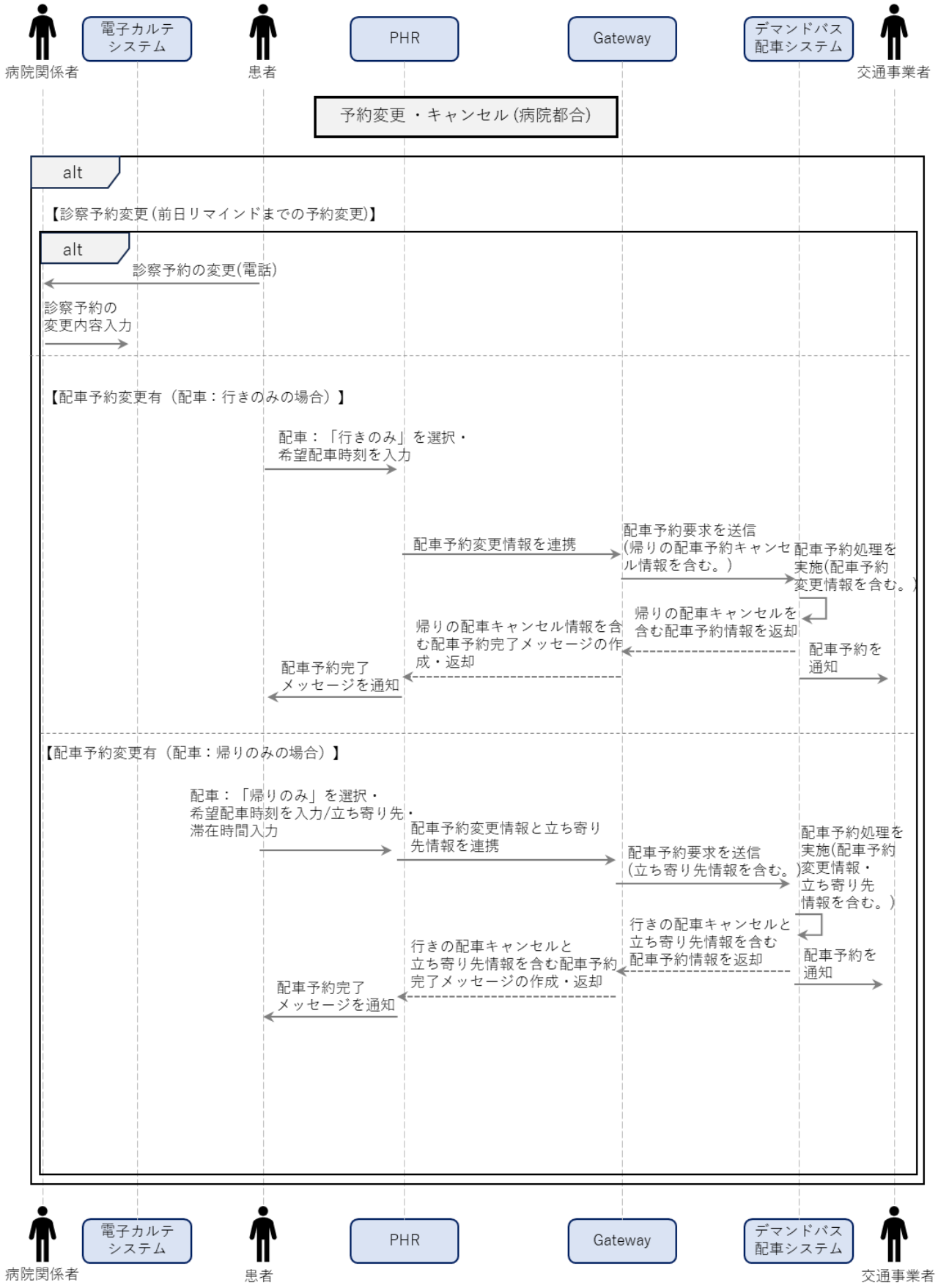


図 1-6　予約変更・キャンセル時(病院都合)の処理

④ 診察終了～帰宅の処理

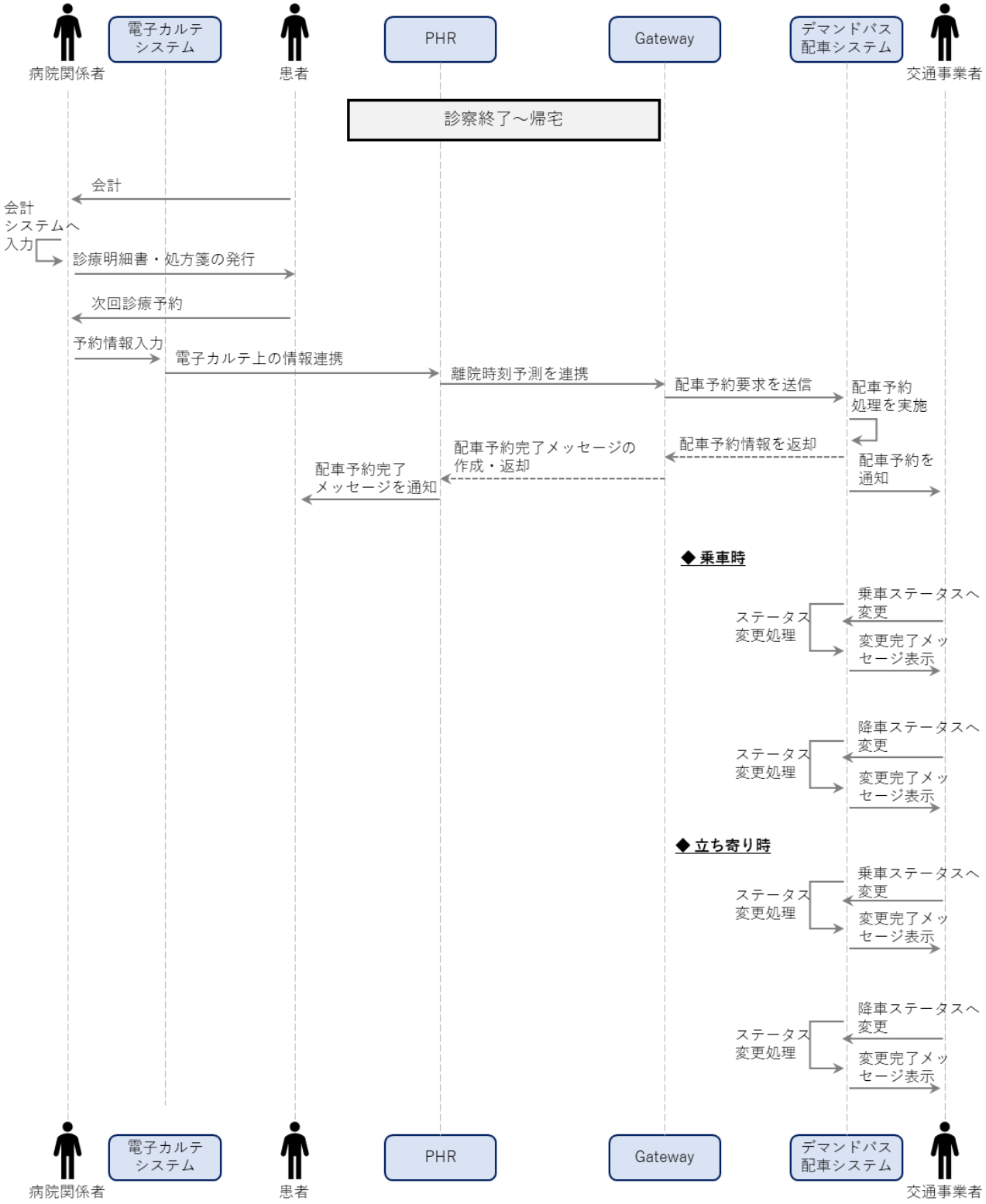


図 1-7 診察終了～帰宅の処理

2. ヘルスケア MaaS 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

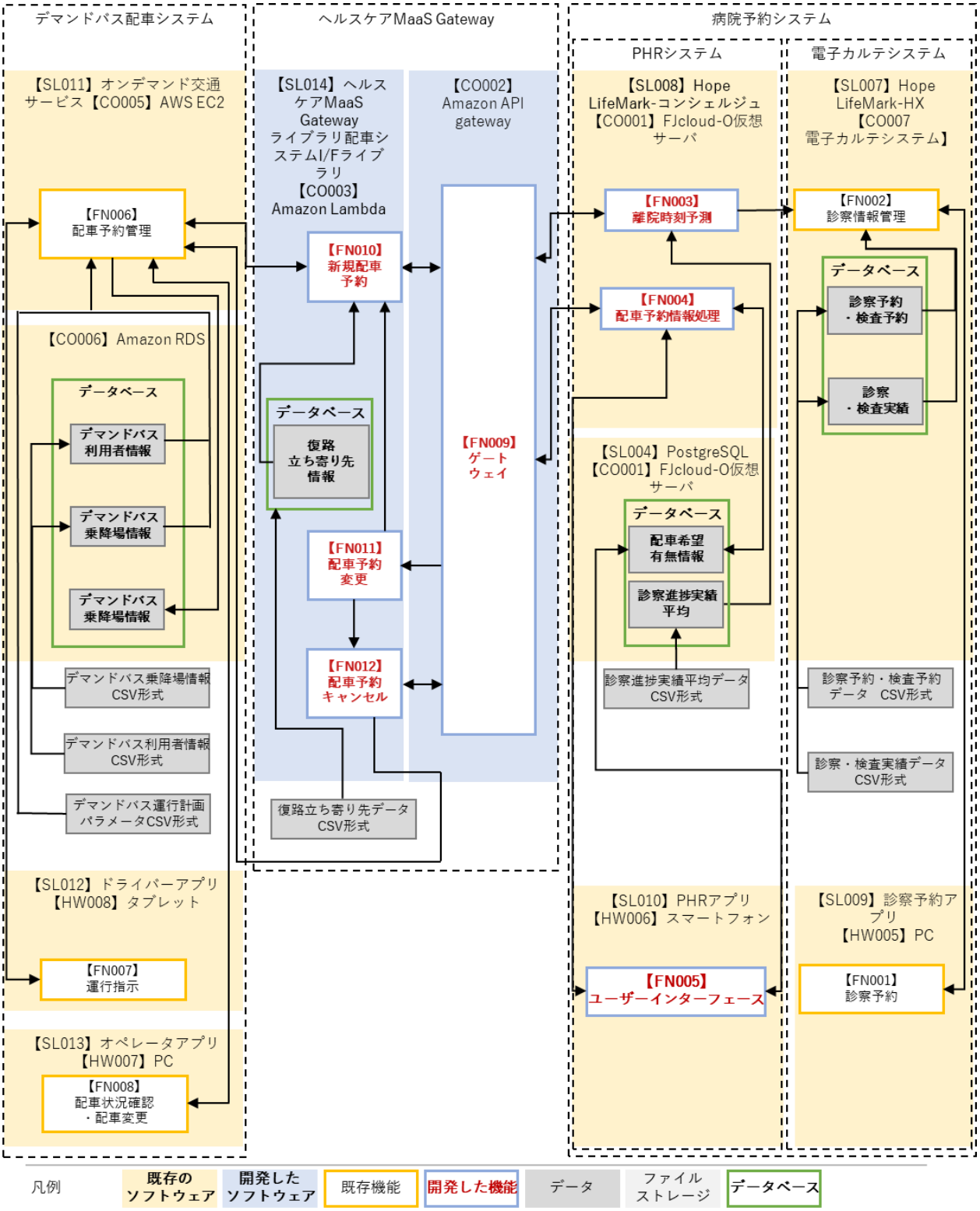


図 2-1 システムアーキテクチャ

2-1-2. システム機能一覧

表 2-1 電子カルテシステム機能一覧

※朱文字：新規開発・既存改修

ID	機能名	機能説明
FN001	診察予約	● 病院関係者等が患者の診察予約情報（患者情報、診察日、診察・検査内容等）を入力する。
FN002	診察情報管理	● 患者情報、診察日、診察・検査内容等に基づき、予約データを電子カルテシステムに保存する。

表 2-2 PHR 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
FN003	離院時刻予測	● 外来診察における離院時刻を離院時刻予測モデルにより次回診察予約時、診察前日、診察当日に予測する。
FN004	配車予約情報処理	● 診察予約時刻、離院予測時刻及び復路の立ち寄り先を基に配車されたデマンドバスの配車予約結果を PHR システムへ格納する。データベース上、配車希望有無が「希望無」の場合はデマンドバスの配車予約を行わない。 ● FN005 から受け取った配車予約キャンセルリクエストを FN009 に送信する。
FN005	ユーザーインターフェース	● スマートフォンにて診察予約と連動した配車情報の照会、キャンセルと立ち寄り先の指定を可能とする。なお、診察予約と連動した配車予約の希望有無の情報を表示、更新可能とする。

表 2-3 デマンドバス配車システム機能一覧

※朱文字：新規開発・既存改修

ID	機能名	機能説明
FN006	配車予約管理	● デマンドバスの配車予約情報を基に空き車両を検索しデマンドバスの配車を行い、デマンドバスのキャンセル情報を基に該当予約のキャンセルを行う。 ● また、これら実績（検索、配車、キャンセル）は蓄積され、実績データとして出力できる。

表 2-4 ドライバーアプリ機能一覧

※朱文字：新規開発・既存改修

ID	機能名	機能説明
FN007	運行指示	● デマンドバスの配車システムにて登録された運行経路を画面表示し、乗降車のステータス更新を可能とする。

表 2-5 オペレータアプリ機能一覧

※朱文字：新規開発・既存改修

ID	機能名	機能説明
FN008	配車状況確認・配車変更	● デマンドバス配車システムの運行状況の照会を可能とし、必要があれば予約の変更やキャンセルを可能とする。

表 2-6 ヘルスケア MaaS Gateway 機能一覧

※朱文字：新規開発・既存改修

ID	機能名	機能説明
FN009	ゲートウェイ	● PHR システムとデマンドバス配車システムの連携を実現するためにデマンドバスの配車予約情報、デマンドバスの配車予約キャンセル情報などを連携する。
FN010	新規配車予約	● 病院到着希望日時、病院出発希望日時及び復路の立ち寄り先、立ち寄り時間を基にデマンドバスの配車予約情報を生成する。
FN011	配車予約変更	● 配車希望日時が変更された場合、該当の配車予約の変更処理を行う。
FN012	配車キャンセル	● 利用者によりデマンドバスの配車予約がキャンセルされた場合、それに紐づく配車予約のキャンセルを行う。

2-1-3. システム機能の詳細

以下に、システム機能の詳細を記す。なお、本業務において開発（新規・改修）を行うシステム機能は、機能名称を朱文字で示す。

【FN001】診察予約

- 本システム機能の概要
  - 電子カルテシステム上で次回診察予約の登録を行う。
- 本システム機能の入力・処理・出力のフローチャート

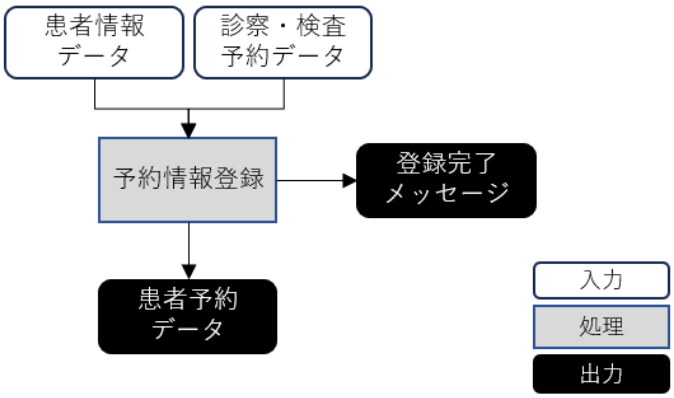


図 2-2 【FN001】のフローチャート

- 本システム機能の処理の詳細
  - 予約情報登録
    - ✧ 処理内容
      - 診療予約アプリ画面から患者カルテを開き、次回診察日、診察内容等を参照し次回診察予約の情報を入力する。
      - 電子カルテシステム上に、入力された診察予約データを【FN002】診療情報管理へ連携する。
    - ✧ 利用するライブラリ
      - 【SL009】診察予約アプリ
    - ✧ 利用するアルゴリズム
      - -
- データ仕様
  - 入力
    - ✧ 患者情報データ
      - データの内容
        - 患者情報
      - データの形式
        - JSON 形式
      - 利用するデータインターフェース

- 【IF001】診療予約アプリ画面入力情報連携 IF

- 診察・検査予約データ
  - データの内容
    - 診察日時、診察・検査予約内容等
  - データの形式
    - JSON 形式
  - 利用するデータインターフェース
    - 【IF001】診療予約アプリ画面入力情報連携 IF

- 出力
  - 患者予約データ、登録完了メッセージ
    - データの内容
      - 患者予約情報
    - データの形式
      - JSON 形式
    - 利用するデータインターフェース
      - 【IF002】診療予約アプリ画面表示情報連携 IF

【FN002】診察情報管理

- 本システム機能の概要
  - 患者情報、診察日、診察・検査内容等に基づき、予約データを電子カルテシステムに保存する 。
- 本システム機能の入力・処理・出力のフローチャート

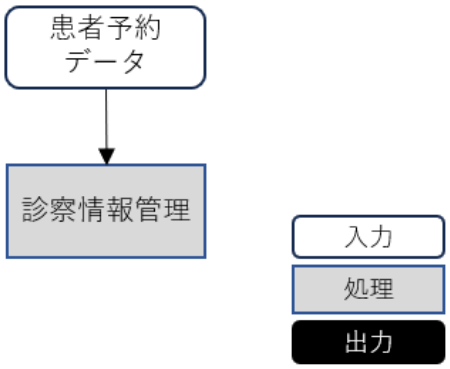


図 2-3【FN002】のフローチャート

- 本システム機能の処理の詳細
  - 診察情報管理
    - 処理内容
      - 【FN001】診察予約で出力された次回診察予約データをインプットに、電子カルテシステムに診察予約情報を格納する。
    - 利用するライブラリ
      - 【SL007】Hope LifeMark-HX
    - 利用するアルゴリズム
      - -
- データ仕様
  - 入力
    - 患者予約データ
      - データの内容
        - 患者予約情報
      - データの形式
        - JSON 形式
      - 利用するデータインターフェース
        - 【IF001】診療予約アプリ画面入力情報連携 IF
  - 出力
    - 患者予約データ
      - データの内容
        - 患者予約情報
      - データの形式

- JSON 形式
- 利用するデータインターフェース
  - 【IF010】診察予約・検査予約データ連携 IF
  - 【IF011】診察進捗実績データ IF

【FN003】離院時刻予測＜新規開発＞

- 本システム機能の概要
  - 外来診察における離院時刻を予測し、デマンドバスの復路配車時刻を算出するため、電子カルテシステムから取得した実績データ等を基に離院予測時刻を出力する。
  - 過去の診察・検査・会計等の実績データからルールベースで離院時刻予測を行う。
- 本システム機能の入力・処理・出力のフローチャート

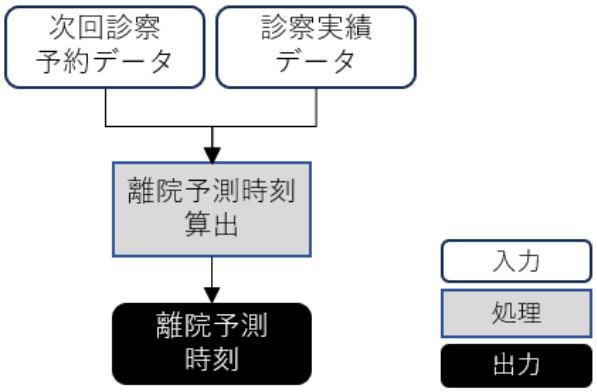


図 2-4【FN003】のフローチャート

- 本システム機能の処理の詳細
  - 離院予測時刻算出
    - 処理内容
      - 外来診察における離院時刻を予測し、デマンドバスの復路配車時刻を算出するため、電子カルテシステムから取得した実績データ等を基に離院予測時刻を FN007 運行指示へ出力する。
      - 離院時刻予測は、定めたルールにのっとって過去実績を基にあらかじめ計算した予測時間（ルールごとの固定値）を足し算することで離院時刻を予測する。
      - 離院予測時刻算出については次回診察予約時、診察前日、診察当日の 3 つの回のバッチ処理のタイミングによって予約内容の変更有無を【FN002】診察情報管理に問合せの上、確認し、変更があった場合は離院時刻を再計算して、【FN009】ゲートウェイへ復路配車時刻の変更をリクエストする。
        ※詳細は【AL101】離院時刻予測モデルを参照
    - 利用するライブラリ
      - 【SL002】Java
      - 【SL005】Apache HTTP Server
      - 【SL006】Apache Tomcat

- 【SL008】Hope LifeMark-コンシェルジュ

✧ 利用するアルゴリズム

- 【AL101】離院時刻予測モデル

- データ仕様
  - 入力
    - ✧ 次回診察予約データ
      - データの内容
        - 患者情報、診察日時、診察・検査内容等
      - データの形式
        - JSON 形式
      - 利用するデータインターフェース
        - 【IF003】離院時刻予測ー診察情報連携 IF
    - ✧ 診察実績データ
      - データ内容
        - 患者情報、診察日時実績、診察・検査実績
      - データの形式
        - JSON 形式
      - 利用するデータインターフェース
        - 【IF003】離院時刻予測ー診察情報連携 IF
  - 出力
    - ✧ 離院予測時刻
      - データの内容
        - 離院予測時刻
      - データの形式
        - JSON 形式
      - 利用するデータインターフェース
        - 【IF005】離院時刻予測ーゲートウェイ連携 IF
        - 【IF012】診察進捗実績平均データ IF

【FN004】配車予約情報処理＜新規開発＞

- 本システム機能の概要
  - 診察予約時刻、離院予測時刻及び復路の立ち寄り先を基に配車されたデマンドバスの配車予約結果を PHR システム上に格納する。なお、PHR アプリで配車予約「希望無」が指定された場合はデマンドバスの配車予約を行わない。
  - PHR アプリで配車予約について、往路、復路、又はその両方がキャンセルされた場合にキャンセル情報をヘルスケア MaaS Gateway へ送信する 。
  - PHR アプリで指定された立ち寄り先情報をヘルスケア Maas Gateway へ送信する。
- 本システム機能の入力・処理・出力のフローチャート

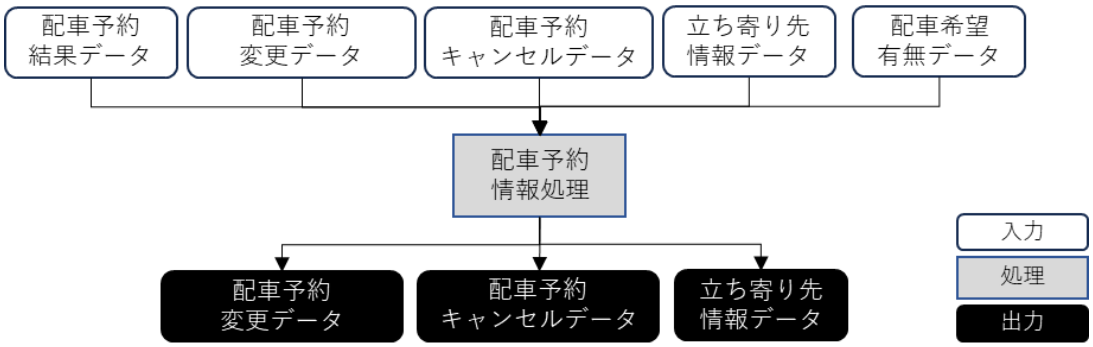


図 2-5【FN004】のフローチャート

- 本システム機能の処理の詳細
  - 配車予約情報処理
    - 処理内容
      - 【FN009】ゲートウェイから受け取ったデマンドバスの配車予約結果を保存する。
    - 利用するライブラリ
      - 【SL002】Java
      - 【SL005】Apache HTTP Server
      - 【SL006】Apache Tomcat
      - 【SL008】Hope LifeMark-コンシェルジュ
    - 利用するアルゴリズム
      - -
  - 配車予約変更送信
    - 処理内容
      - 【FN005】ユーザーインターフェースから受け取った配車予約変更リクエストを【FN009】ゲートウェイに送信する。データベース上、配車希望有無が「希望無」の場合は配車予約変更リクエストを送信しない。
    - 利用するライブラリ
      - 【SL002】Java
      - 【SL008】Hope LifeMark-コンシェルジュ
    - 利用するアルゴリズム
      - -
  - 配車予約キャンセル送信
    - 処理内容
      - 【FN005】ユーザーインターフェースから受け取った配車予約キャンセルリクエストを【FN009】ゲートウェイに送信する。
    - 利用するライブラリ
      - 【SL002】Java
      - 【SL008】Hope LifeMark-コンシェルジュ
    - 利用するアルゴリズム
      - -

- 立ち寄り先情報送信
  - 処理内容
    - 【FN005】ユーザーインターフェースから受け取った立ち寄り情報を【FN009】ゲートウェイに送信する。
  - 利用するライブラリ
    - 【SL002】Java
    - 【SL008】Hope LifeMark-コンシェルジュ
  - 利用するアルゴリズム
    - -

- データ仕様
  - 入力
    - 配車予約結果データ
      - データの内容
        - 【FN009】ゲートウェイから受け取ったデマンドバスの配車予約結果データ
      - データの形式
        - JSON 形式
      - 利用するデータインターフェース
        - 【IF004】配車予約検索確定標準 API
    - 配車予約変更データ
      - データの内容
        - 【FN005】ユーザーインターフェースから受け取った配車予約変更データ
      - データの形式
        - JSON 形式
      - 利用するデータインターフェース
        - 【IF008】PHR アプリ画面入力情報連携 IF
    - 配車予約キャンセルデータ
      - データの内容
        - 【FN005】ユーザーインターフェースから受け取った配車予約キャンセルデータ
      - データの形式
        - JSON 形式
      - 利用するデータインターフェース
        - 【IF008】PHR アプリ画面入力情報連携 IF
    - 立ち寄り先情報データ
      - データの内容
        - 【FN005】ユーザーインターフェースから受け取った立ち寄り情報
      - データの形式
        - JSON 形式
      - 利用するデータインターフェース

- 【IF008】PHR アプリ画面入力情報連携 IF

✧ 配車希望有無データ

- データの内容
  - データベース上の配車希望有無データから取得した配車希望有無情報
- データの形式
  - SQL 形式
- 利用するデータインターフェース
  - 【IF034】配車希望有無取得 IF

➢ 出力

✧ 配車予約変更データ

- データの内容
  - 配車予約変更情報
- データの形式
  - JSON 形式
- 利用するデータインターフェース
  - 【IF006】配車予約変更標準 API

✧ 配車予約キャンセルデータ

- データの内容
  - 配車キャンセル情報
- データの形式
  - JSON 形式
- 利用するデータインターフェース
  - 【IF007】配車予約キャンセル標準 API

✧ 立ち寄り先情報データ

- データの内容
  - 立ち寄り先場所、立ち寄り先滞在時間
- データの形式
  - JSON 形式
- 利用するデータインターフェース
  - 【IF006】配車予約変更標準 API

【FN005】ユーザーインターフェース＜新規開発＞

- 本システム機能の概要
  - ➢ デマンドバス配車システムで予約された配車予約情報を表示する。
  - ➢ デマンドバス配車システムで予約された配車予約をキャンセルする。
  - ➢ 復路の立ち寄り先を指定する。
  - ➢ なお、診察予約と連動した配車予約の希望有無の情報を表示、更新可能とする。
- 本システム機能の入力・処理・出力のフローチャート

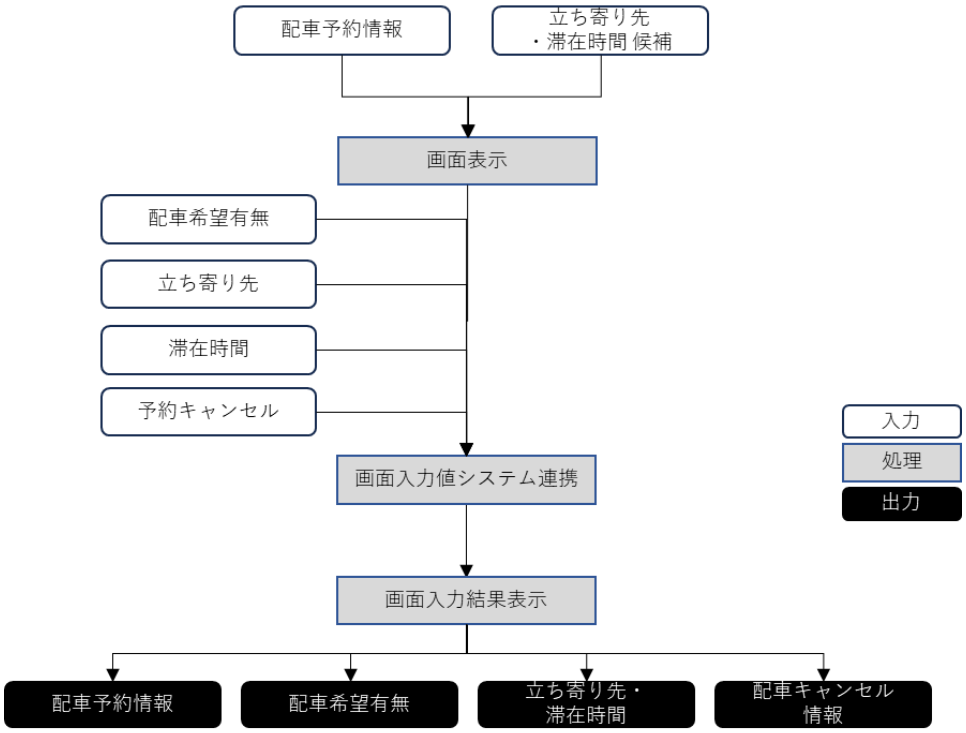


図 2-6【FN005】のフローチャート

- 本システム機能の処理の詳細
  - 配車予約日時表示
    - 処理内容
      - 【FN004】配車予約情報処理から受け取った配車予約日時・立ち寄り先を表示する。
    - 利用するライブラリ
      - 【SL002】 Java
    - 利用するアルゴリズム
      - -
  - 配車希望有無
    - 処理内容
      - 【FN005】診察予約に基づく配車の希望有無を表示、更新する。
    - 利用するライブラリ
      - 【SL004】 PostgreSQL
    - 利用するアルゴリズム
      - -
  - 立ち寄り先指定
    - 処理内容
      - 復路に立ち寄りたい移動先を移動先リスト表示し、指定された立ち寄り先情報を【FN004】配車予約情報処理へ送信する。
    - 利用するライブラリ
      - 【SL002】Java

- 【SL003】Swift
- 【SL010】PHR アプリ

✧ 利用するアルゴリズム

- -

➢ 配車キャンセル

✧ 処理内容

- 配車予約情報をキャンセルし、配車キャンセル情報を FN004 へ送信する。

✧ 利用するライブラリ

- 【SL002】Java
- 【SL003】Swift
- 【SL010】PHR アプリ

✧ 利用するアルゴリズム

- -

- データ仕様

➢ 入力

✧ 配車予約情報

- データの内容
  - 【FN004】配車予約情報処理から受け取った配車予約日時等の情報
- データの形式
  - SQL 形式
- 利用するデータインターフェース
  - 【IF009】PHR アプリ画面表示情報連携 IF

✧ 配車希望有無データ

- データの内容
  - データベース上の配車希望有無データから取得した配車希望有無情報
- データの形式
  - SQL 形式
- 利用するデータインターフェース
  - 【IF034】配車希望有無取得 IF

✧ 立ち寄り先・滞在時間候補

- データの内容
  - 【FN005】ユーザーインターフェースから受け取った立ち寄り情報
- データの形式
  - JSON 形式
- 利用するデータインターフェース
  - 【IF009】PHR アプリ画面表示情報連携 IF

➢ 出力

✧ 配車予約情報

- データの内容
  - 配車予約情報
- データの形式
  - JSON 形式
- 利用するデータインターフェース
  - 配車予約日時
  - 立ち寄り先場所名称
  - 立ち寄り先滞在時間

✧ 配車希望有無

- データの内容
  - 【FN005】診察予約に基づく配車の希望有無を表示する。
- データの形式
  - JSON 形式
- 利用するデータインターフェース
  - 【IF033】配車希望有無入力 IF
  - 【IF033】配車希望有無取得 IF

✧ 立ち寄り先・滞在時間

- データの内容
  - 立ち寄り先場所、立ち寄り先滞在時間
- データの形式
  - JSON 形式
- 利用するデータインターフェース
  - 【IF008】PHR アプリ画面入力情報連携 IF

✧ 配車予約変更情報

- データの内容
  - 配車予約変更情報
- データの形式
  - JSON 形式
- 利用するデータインターフェース
  - 【IF008】PHR アプリ画面入力情報連携 IF

✧ 配車キャンセル情報

- データの内容
  - 配車キャンセル情報
- データの形式
  - JSON 形式
- 利用するデータインターフェース
  - 【IF008】PHR アプリ画面入力情報連携 IF

【FN006】配車予約管理

- 本システム機能の概要
  - デマンドバスの配車予約情報を基に空き車両を検索しデマンドバスの配車を行い、デマンドバスのキャンセル情報を基に該当予約のキャンセルを行う。また、これら実績（検索、配車、キャンセル）は蓄積され、実績データとして出力できる。
- 本システム機能の入力・処理・出力のフローチャート

【FN006】配車予約管理

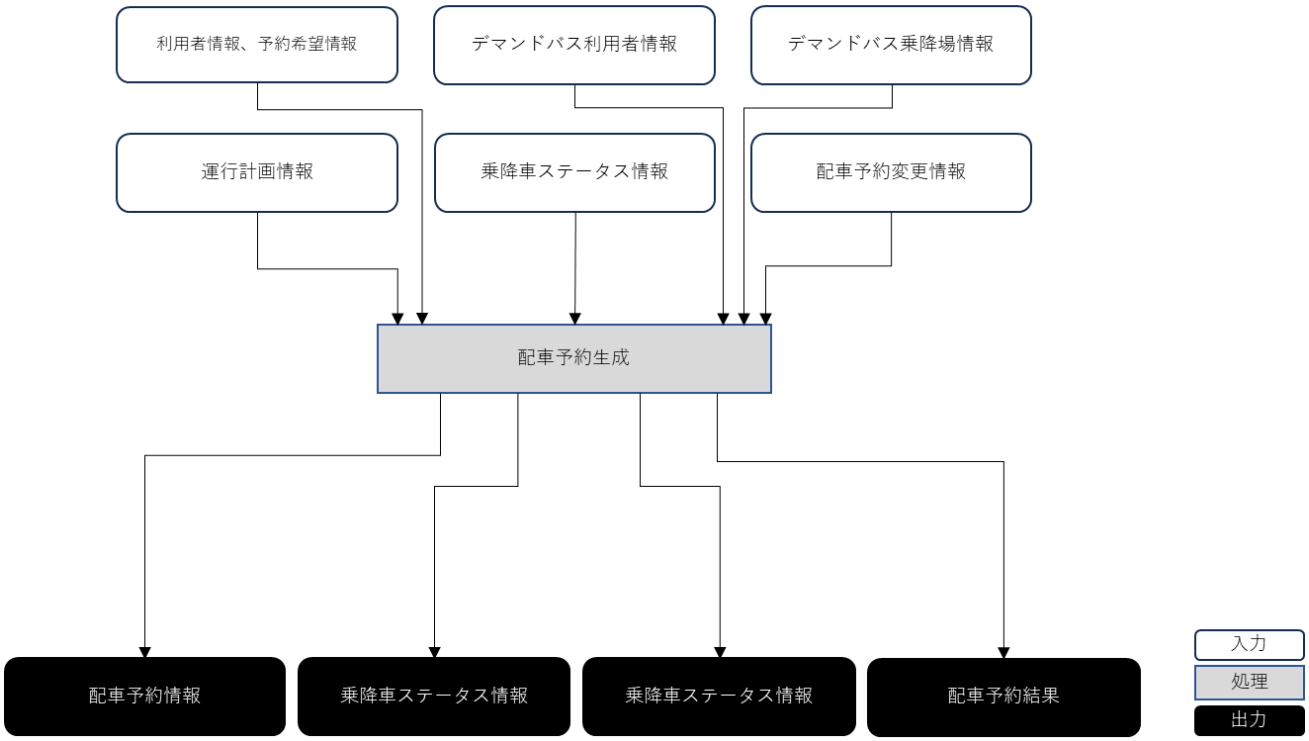


図 2-7【FN006】のフローチャート

- 本システム機能の処理の詳細
  - 予約機能
    - 処理内容
      - 外部連携 API から配車予約情報を受信し、配車指示を行う。
    - 利用するライブラリ
      - 【SL011】デマンド交通サービス
    - 利用するアルゴリズム
      - -
- データ仕様
  - 入力
    - 利用者情報、予約希望情報
      - データの内容
        - 【IF024】予約一覧取得標準 API から連携される利用者と予約に関する情報

- 【IF025】デマンドバス配車予約検索標準 API から連携される予約候補をリストアップするための検索条件に関する情報
- 【IF026】デマンドバス配車 API から連携される予約確定させるために必要な情報

- データの形式
  - XML 形式
- 利用するデータインターフェース
  - 【IF024】予約一覧取得標準 API
  - 【IF025】デマンドバス配車予約検索標準 API
  - 【IF026】デマンドバス配車 API

✧ デマンドバス利用者情報

- データの内容
  - 【IF013】デマンドバス利用者情報連携 IF から連携される利用者情報
- データの形式
  - REST API 形式
- 利用するデータインターフェース
  - 【IF013】デマンドバス利用者情報連携 IF

✧ デマンドバス乗降場情報

- データの内容
  - 【IF015】デマンドバス乗降場情報連携 IF から連携されるデマンド乗降場情報
- データの形式
  - REST API 形式
- 利用するデータインターフェース
  - 【IF015】デマンドバス乗降場情報連携 IF

✧ 運行計画情報

- データの内容
  - 【IF017】デマンドバス運行計画パラメータ反映 IF から連携される運行計画情報
- データの形式
  - CSV 形式
- 利用するデータインターフェース
  - 【IF017】デマンドバス運行計画パラメータ反映 IF

✧ 乗降車ステータス情報

- データの内容
  - 【IF019】ドライバー運行ステータス反映 IF から連携される乗降者ステータス
- データの形式
  - Key-Value 形式
- 利用するデータインターフェース
  - 【IF019】ドライバー運行ステータス反映 IF

- ✧ 配車予約変更情報
  - データの内容
    - 【IF021】オペレータ配車変更反映 IF から連携される配車予約変更情報
  - データの形式
    - XML 形式
  - 利用するデータインターフェース
    - 【IF021】オペレータ配車変更反映 IF

- 出力
  - ✧ 【IF026】デマンドバス配車予約標準 API
    - データの内容
      - 利用者情報、配車予約情報、乗降場情報、乗降時間
    - データの形式
      - REST API 形式
    - 利用するデータインターフェース
      - 【IF026】デマンドバス配車予約標準 API
  - ✧ 【IF018】デマンドバス運行実績出力 IF
    - データの内容
      - 利用者情報、運行実績
    - データの形式
      - CSV 形式
    - 利用するデータインターフェース
      - 【IF018】デマンドバス運行実績出力 IF
  - ✧ 【IF020】ドライバー運行指示 IF
    - データの内容
      - 乗降車ステータス情報
    - データの形式
      - XML 形式
    - 利用するデータインターフェース
      - 【IF020】ドライバー運行指示 IF
  - ✧ 【IF022】オペレータ配車情報反映 IF
    - データの内容
      - 配車予約結果
    - データの形式
      - XML 形式
    - 利用するデータインターフェース
      - 【IF022】オペレータ配車情報反映 IF

【FN007】運行指示

- 本システム機能の概要
  - デマンドバスの配車システムにて登録された運行経路を画面表示する機能である。
  - 指示に従って乗降車がなされたかのステータス更新も可能とする。
- 本システム機能の入力・処理・出力のフローチャート

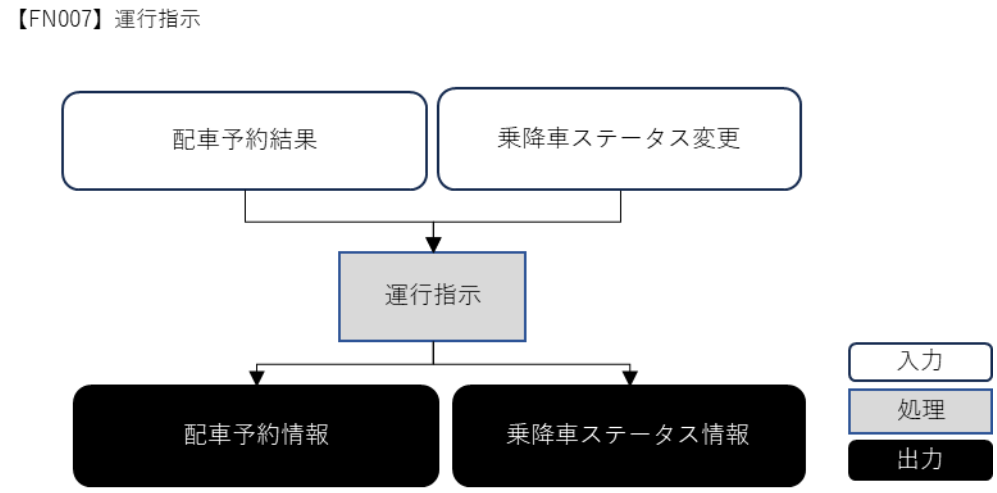


図 2-8【FN007】のフローチャート

- 本システム機能の処理の詳細
  - 運行指示
    - 処理内容
      - 【IF020】ドライバー運行指示 IF を介して【FN006】配車予約管理で生成された運行予約結果を取得して【HW008】ドライバーアプリ画面上に表示し、【HW008】ドライバーアプリ画面から入力された乗降車ステータスを【IF019】ドライバー運行ステータス反映 IF を介して【FN006】配車予約管理に送信する。
    - 利用するライブラリ
      - 【SL012】ドライバーアプリ
    - 利用するアルゴリズム
      - -
- データ仕様
  - 入力
    - 乗車ステータス変更
      - データの内容
        - 乗降車ステータス情報
      - データの形式
        - XML 形式
      - 利用するデータインターフェース
        - 【IF019】ドライバー運行ステータス反映 IF
    - 配車予約結果

- データの内容
  - 【IF020】ドライバー運行指示 IF から連携される配車予約結果
- データの形式
  - XML 形式
- 利用するデータインターフェース
  - 【IF020】ドライバー運行指示 IF

➢ 出力

✧ 【IF019】ドライバー運行ステータス反映 IF

- データの内容
  - 乗降車ステータス情報
- データの形式
  - XML 形式
- 利用するデータインターフェース
  - 【IF019】ドライバー運行ステータス反映 IF

【FN008】配車状況確認・配車変更

- 本システム機能の概要
  - デマンドバス配車システムの運行状況の照会を可能とし、必要があれば電話連絡にて予約の変更やキャンセルを可能とする。
- 本システム機能の入力・処理・出力のフローチャート

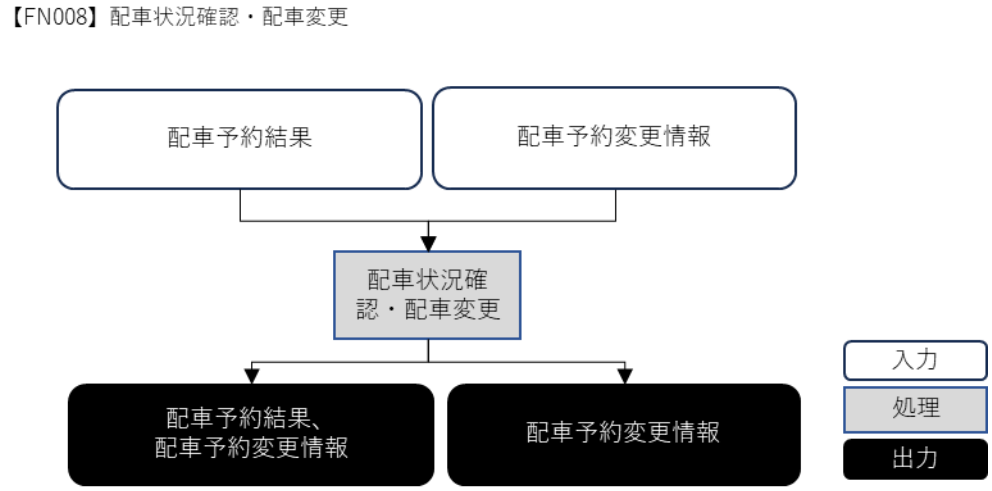


図 2-9【FN008】のフローチャート

- 本システム機能の処理の詳細
  - 配車状況確認・配車変更
    - 処理内容
      - 【FN006】で出力された配車予約結果を【IF022】オペレータ配車情報反映 IF からインプットし、【HW005】PC にて確認し変更があれば【FN008】で入力される予約変更

情報を【IF022】オペレータ配車変更反映 IF を介し【FN006】に出力する。

- ✧ 利用するライブラリ
  - 【SL013】オペレータアプリ
- ✧ 利用するアルゴリズム
  - -

- データ仕様
  - ➢ 入力
    - ✧ 予約確認・変更
      - データの内容
        - ➢ 配車予約結果、配車予約変更情報
      - データの形式
        - ➢ XML 形式
      - 利用するデータインターフェース
        - ➢ 【IF021】オペレータ配車変更反映 IF
    - ✧ 配車予約結果
      - データの内容
        - ➢ 配車予約結果
      - データの形式
        - ➢ XML 形式
      - 利用するデータインターフェース
        - ➢ 【IF021】オペレータ配車変更反映 IF
  - ➢ 出力
    - ✧ 【IF022】オペレータ配車情報反映 IF
      - データの内容
        - ➢ 配車予約変更情報
      - データの形式
        - ➢ XML 形式
      - 利用するデータインターフェース
        - ➢ 【IF022】オペレータ配車情報反映 IF

【FN009】ゲートウェイ<新規開発>

- 本システム機能の概要
  - ➢ PHR システムとデマンドバス配車システムの連携を実現するためにデマンドバスの配車予約情報、デマンドバスの配車予約キャンセル情報などを連携する。
  - ➢ PHR システムとデマンドバス配車システムを直接連携ではなく中間層に本機能を入れる理由としては、連携機能を PHR システム又はデマンドバス配車システムに内包することでベンダー依存となることを避けることと、将来的には病院以外の目的地施設（商業施設等）にも対応するため

- PHR システムから連携される利用者にひも付く離院予測時刻、立ち寄り先情報、配車キャンセル情報を基に、デマンドバス配車システムの配車処理インプットとなる配車リクエストのパラメータに整形することを当機能で行う。なお、配車リクエストは一様ではなく入力データに応じて新規配車予約、配車予約変更、配車キャンセルそれぞれに適切にディスパッチされる。

- 本システム機能の入力・処理・出力のフローチャート

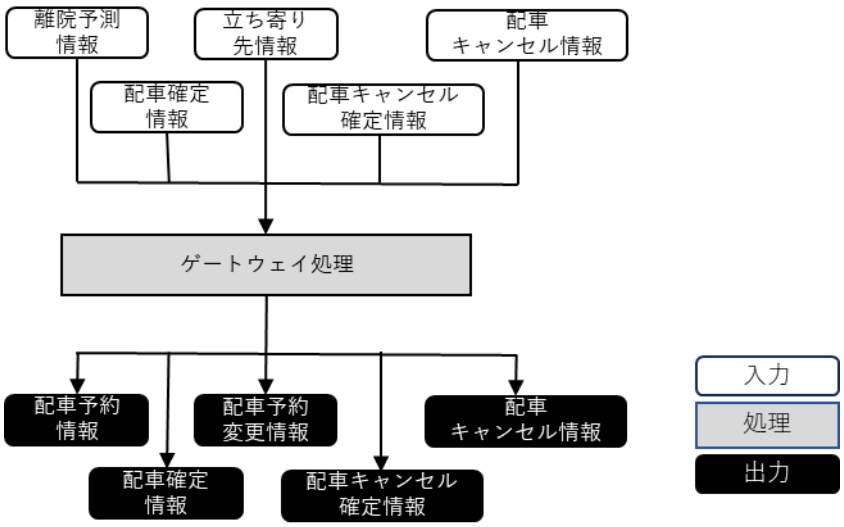


図 2-10【FN009】のフローチャート

- 本システム機能の処理の詳細
  - ゲートウェイ処理
    - 処理内容
      - 【FN003】離院予測時刻から患者ごとの病院到着希望日時と病院出発希望日時を受信する。
      - 【FN004】配車予約情報処理から利用者が【FN005】ユーザーインターフェースにて指定した立ち寄り先情報および滞在時間を受信する。

      ※業務ロジック自体は【FN010】新規配車予約、【FN011】配車予約変更、【FN012】配車予約キャンセルにて実装し、本機能は API 管理を行う（【C002】API サーバーにて実装する。）
    - 利用するライブラリ
      - 【SL014】ヘルスケア MaaS Gateway ライブラリ
    - 利用するアルゴリズム
      - -
- データ仕様
  - 入力
    - 離院時刻予測情報
      - データの内容
        - 患者 ID、病院到着希望日時、病院出発希望日時
      - データの形式

        - REST-API 形式
    - 利用するデータインターフェース
        - 【IF005】離院時刻予測ーゲートウェイ連携 IF
- 立ち寄り先情報
    - データの内容
        - 立ち寄り先コード、立ち寄り先滞在時間
    - データの形式
        - REST-API 形式
    - 利用するデータインターフェース
        - 【IF027】ゲートウェイー配車予約変更連携 IF を参照
- 配車キャンセル情報
    - データの内容
        - 配車予約 ID
    - データの形式
        - Key-Value 形式
    - 利用するデータインターフェース
        - 【IF030】ゲートウェイー配車予約キャンセル連携 IF を参照
- 配車確定情報
    - データの内容
        - 利用者番号、乗車場所（病院、立ち寄り先または自宅)、降車場所（病院、立ち寄り先または自宅)、発車時刻、到着時刻、車両番号
    - データの形式
        - Key-Value 形式
    - 利用するデータインターフェース
        - 【IF023】ゲートウェイー新規配車予約連携 IF を参照
- 配車キャンセル確定情報
    - データの内容
        - 配車予約 ID
    - データの形式
        - Key-Value 形式
    - 利用するデータインターフェース
        - 【IF030】ゲートウェイー配車予約キャンセル連携 IF を参照

- 出力
    - 配車確定情報
        - データの内容
            - 【FN004】配車予約情報処理への処理結果（配車情報）
        - データの形式
            - REST-API 形式

- 利用するデータインターフェース
  - 【IF006】配車予約変更標準 API

✧ 配車キャンセル確定情報

- データの内容
  - 【FN004】配車予約情報処理への処理結果（配車キャンセル情報）
- データの形式
  - Key-Value 形式
- 利用するデータインターフェース
  - 【IF007】配車予約キャンセル標準 API

✧ 配車予約情報

- データの内容
  - 【FN010】新規配車予約へのデータ連携（配車リクエストに必要となる情報）
- データの形式
  - Key-Value 形式
- 利用するデータインターフェース
  - 【IF023】ゲートウェイー新規配車予約連携 IF

✧ 配車予約変更情報

- データの内容
  - 【FN011】配車予約変更へのデータ連携（配車変更リクエストに必要となる情報）
- データの形式
  - Key-Value 形式
- 利用するデータインターフェース
  - 【IF027】ゲートウェイー配車予約変更連携 IF

✧ 配車キャンセル情報

- データの内容
  - 【FN012】配車予約キャンセルへのデータ連携（配車キャンセルリクエストに必要となる情報）
- データの形式
  - Key-Value 形式
- 利用するデータインターフェース
  - 【IF030】ゲートウェイー配車予約キャンセル連携 IF

【FN010】新規配車予約<新規開発>

- 本システム機能の概要
  - 病院到着希望日時、病院出発希望日時及び復路の立ち寄り先、立ち寄り時間を基にデマンドバスの配車予約情報を生成し、デマンドバス配車システムで配車予約検索を行い、予約候補を受け取る。予約候補の中から最適な車両を選択し、その車両の予約確定を行う。予約確定がなされた後、当該予約確定情報を PHR システムへ返す。

- 本システム機能の入力・処理・出力のフローチャート

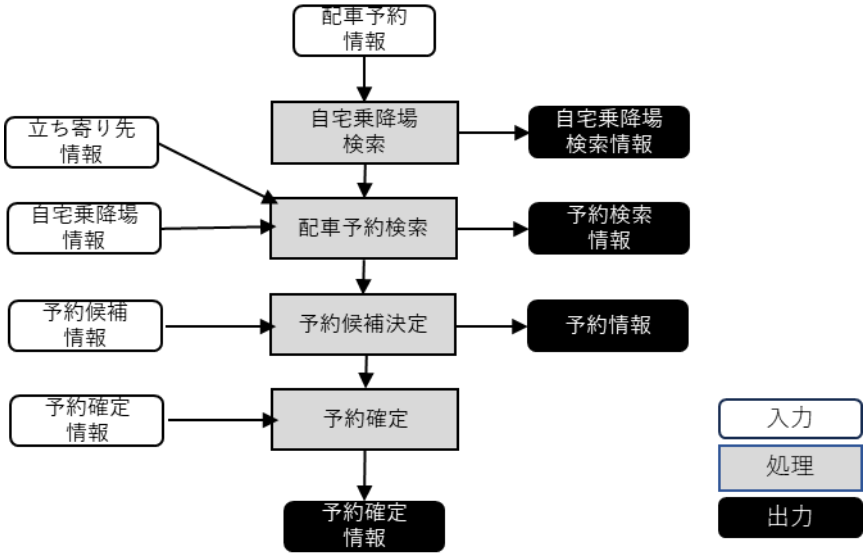


図 2-11【FN010】のフローチャート

- 本システム機能の処理の詳細
  - 自宅乗降場検索
    - 処理内容
      - 次回診察予約時に PHR システムから連携された利用者の自宅の場所をデマンドバス配車システムに問合せ特定する。
    - 利用するライブラリ
      - 【SL001】Node.js
      - 【SL014】ヘルスケア MaaS Gateway ライブラリ
    - 利用するアルゴリズム
      - -
  - 新規配車予約検索
    - 処理内容
      - 次回診察予約時に PHR システムから連携された患者ごとの病院到着希望日時、病院出発希望日時、自宅乗降場及び立ち寄り先を基に新規配車リクエストをデマンドバス配車サービスに発行する。
    - 利用するライブラリ
      - 【SL001】Node.js
      - 【SL014】ヘルスケア MaaS Gateway ライブラリ
    - 利用するアルゴリズム
      - -
  - 予約候補決定
    - 処理内容
      - 新規配車予約検索処理の結果としてデマンドバス配車システムから得られた予約候補の情報から、希望時刻に最も近い時間の予約候補の予約 ID を予約確定リクエストとして

デマンドバス配車サービスに発行する。

- ✧ 利用するライブラリ
  - 【SL001】Node.js
  - 【SL014】ヘルスケア MaaS Gateway ライブラリ
- ✧ 利用するアルゴリズム
  - -

➢ 予約確定

- ✧ 処理内容
  - 予約候補決定処理の結果としてデマンドバス配車システムから得られた予約情報を PHR アプリ画面に反映させるべく【FN009】に連携する。
- ✧ 利用するライブラリ
  - 【SL001】Node.js
  - 【SL014】ヘルスケア MaaS Gateway ライブラリ
- ✧ 利用するアルゴリズム
  - -

- データ仕様

➢ 入力

- ✧ 配車予約情報
  - データの内容
    - ➢ 配車リクエストに必要となる情報（利用者番号、乗車場所（病院、立ち寄り先または自宅)、降車場所（病院、立ち寄り先または自宅)、発車時刻、到着時刻）
  - データの形式
    - ➢ Key-Value 形式
  - 利用するデータインターフェース
    - ➢ 【IF023】ゲートウェイー新規配車予約連携 IF
    - ➢ 【IF029】配車予約変更―新規配車予約連携 IF
- ✧ 自宅乗降場情報
  - データの内容
    - ➢ 自宅乗降場のコード
  - データの形式
    - ➢ REST-API 形式
  - 利用するデータインターフェース
    - ➢ 【IF024】予約一覧取得標準 API
- ✧ 予約候補情報
  - データの内容
    - ➢ デマンドバス配車システムから得られた予約候補情報（利用者番号、乗車場所（病院、立ち寄り先または自宅)、降車場所（病院、立ち寄り先または自宅)、発車時刻、到着時刻、車両番号）

- データの形式
  - REST-API 形式
- 利用するデータインターフェース
  - 【IF025】デマンドバス配車予約検索標準 API

✧ 予約確定情報

- データの内容
  - デマンドバス配車システムから得られた予約確定情報（利用者番号、乗車場所（病院、立ち寄り先または自宅）、降車場所（病院、立ち寄り先または自宅）、発車時刻、到着時刻、車両番号）
- データの形式
  - REST-API 形式
- 利用するデータインターフェース
  - 【IF026】デマンドバス配車予約標準 API

➢ 出力

✧ 自宅乗降場検索情報

- データの内容
  - 利用者番号（患者 ID を設定）、自宅乗降場コード（デマンドバス配車システムに登録済情報）
- データの形式
  - REST API 形式
- 利用するデータインターフェース
  - 【IF024】予約一覧取得標準 API

✧ 予約検索情報

- データの内容
  - 利用者番号（患者 ID を設定）、乗車場所（病院、自宅または立ち寄り先）、降車場所（病院、自宅または立ち寄り先）、発車時刻、到着時刻
- データの形式
  - REST API 形式
- 利用するデータインターフェース
  - 【IF025】デマンドバス配車予約検索標準 API

✧ 予約情報

- データの内容
  - 配車予約 ID、利用者番号（患者 ID を設定）
- データの形式
  - REST API 形式
- 利用するデータインターフェース
  - 【IF026】デマンドバス配車予約標準 API

✧ 予約確定情報

- データの内容
  - 配車予約 ID、利用者番号（患者 ID を設定）、乗車場所（病院、自宅または立ち寄り先）、降車場所（病院,、自宅または立ち寄り先）、発車時刻、到着時刻
- データの形式
  - Key-Value 形式
- 利用するデータインターフェース
  - 【IF009】PHR アプリ画面表示情報連携 IF

【FN011】配車予約変更<新規開発>

- 本システム機能の概要
  - 変更された病院到着希望日時、病院出発希望日時及び復路の立ち寄り先、立ち寄り時間を基にデマンドバスの配車予約情報を変更する
- 本システム機能の入力・処理・出力のフローチャート

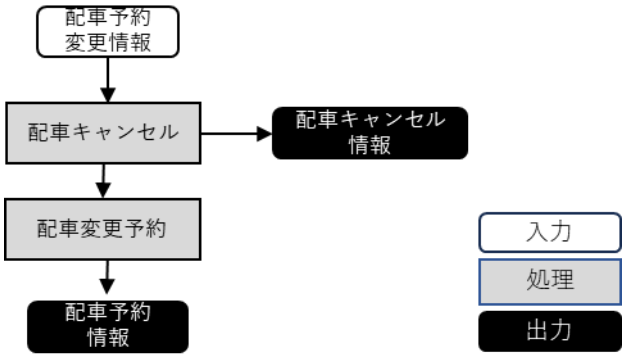


図 2-12【FN011】のフローチャート

- 本システム機能の処理の詳細
  - 配車キャンセル
    - 処理内容
      - 既に登録されている配車予約を削除してから登録させるため配車予約キャンセルリクエストパラメータの生成および配車予約キャンセルリクエスト実行を最初に行う。処理としては【FN012】配車予約キャンセルと共通となる。
    - 利用するライブラリ
      - 【SL001】Node.js
      - 【SL014】ヘルスケア MaaS Gateway ライブラリ
    - 利用するアルゴリズム
      - -
  - 配車変更予約
    - 処理内容
      - 診察リマインドまたは当日受付チェックイン時に PHR システムから連携された患者ごとの病院到着希望日時、病院出発希望日時および立ち寄り先情報（指定がある場合）を基に

新規配車リクエストパラメータを生成する。処理としては【FN010】新規配車予約と共通となる。

- ✧ 利用するライブラリ
  - 【SL001】Node.js
  - 【SL014】ヘルスケア MaaS Gateway ライブラリ
- ✧ 利用するアルゴリズム
  - -

- データ仕様
  - ➢ 入力
    - ✧ 配車予約変更
      - データの内容
        - ➢ 配車予約 ID、患者 ID、病院到着希望日時、病院出発希望日時、立ち寄り先コード、滞在時間
      - データの形式
        - ➢ Key-Value 形式
      - 利用するデータインターフェース
        - ➢ 【IF027】ゲートウェイー配車予約変更連携 IF
  - ➢ 出力
    - ✧ 配車キャンセル情報
      - データの内容
        - ➢ 配車予約 ID、患者 ID
      - データの形式
        - ➢ Key-Value 形式
      - 利用するデータインターフェース
        - ➢ 【IF028】配車予約変更―配車キャンセル連携 IF
    - ✧ 配車予約情報
      - データの内容
        - ➢ 配車リクエストに必要となる情報（利用者番号、乗車場所（病院、立ち寄り先または自宅)、降車場所（病院、立ち寄り先または自宅)、発車時刻、到着時刻）
      - データの形式
        - ➢ Key-Value 形式
      - 利用するデータインターフェース
        - ➢ 【IF029】配車予約変更―新規配車予約連携 IF

【FN012】配車予約キャンセル<新規開発>

- 本システム機能の概要
  - ➢ 配車予約キャンセル情報を基にデマンドバスの配車予約キャンセルのリクエストを生成する 。
- 本システム機能の入力・処理・出力のフローチャート

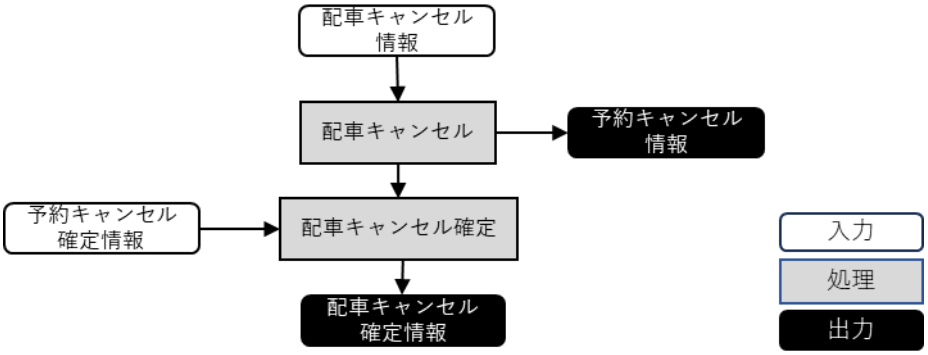


図 2-13【FN012】のフローチャート

- 本システム機能の処理の詳細
  - 配車キャンセル
    - 処理内容
      - PHR アプリ画面にて配車キャンセルが行われた際に、対象となる利用者と予約情報をキーとして配車キャンセルパラメータを生成しデマンドバス配車システムに配車キャンセルリクエストを発行する。
    - 利用するライブラリ
      - 【SL001】Node.js
      - 【SL014】ヘルスケア MaaS Gateway ライブラリ
    - 利用するアルゴリズム
      - -
  - 配車キャンセル確定
    - 処理内容
      - デマンドバス配車システムにて実行された配車予約キャンセル結果を PHR アプリ画面に反映すべく、【FN009】ゲートウェイに連携する。
    - 利用するライブラリ
      - 【SL001】Node.js
      - 【SL014】ヘルスケア MaaS Gateway ライブラリ
    - 利用するアルゴリズム
      - -
- データ仕様
  - 入力
    - 配車キャンセル情報
      - データの内容
        - 患者 ID、キャンセル対象予約 ID
      - データの形式
        - Key-Value 形式

- 利用するデータインターフェース
  - 【IF028】配車予約連行―配車キャンセル連携 IF
  - 【IF030】ゲートウェイー配車予約キャンセル連携 IF

✧ 配車予約キャンセル確定情報

- データの内容
  - 患者 ID、キャンセル対象予約 ID
- データの形式
  - REST API 形式
- 利用するデータインターフェース
  - 【IF031】デマンドバス配車予約キャンセル標準 API

➢ 出力

✧ 配車予約キャンセル情報

- データの内容
  - キャンセル対象となる予約 ID をキーとした配車予約キャンセルパラメータを生成する。
- データの形式
  - REST API 形式
- 利用するデータインターフェース
  - 【IF031】デマンドバス配車予約キャンセル標準 API

✧ 配車キャンセル確定情報

- データの内容
  - 患者 ID、キャンセル対象予約 ID
- データの形式
  - Key-Value 形式
- 利用するデータインターフェース
  - 【IF030】ゲートウェイー配車予約キャンセル連携 IF

2-1-4. ソフトウェア・ライブラリ（SL）の詳細

表 2-7 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	Node.js	LTS24	● JavaScript の実行環境である。 ● ヘルスケア MaaS Gateway に実装する機能を記述するに利用する。
SL002	Java	Java 21	● オブジェクト指向のプログラミング言語である。 ● PHR システムおよび PHR アプリ（Android）におけるプログラミングに適用する。
SL003	Swift	Swift6.1	● 主に Apple のエコシステム（iOS、macOS、watchOS、tvOS）向けのプログラミング言語である。 ● PHR アプリ（iOS）におけるプログラミングに適用する。
SL004	PostgreSQL	PostgreSQL17.5	● 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である。 ● PHR システムにおける利用者情報及び離院時刻予測結果算出用平均実績値等の管理に適用する。
SL005	Apache HTTP Server	Apache HTTP Server. 2.4.63	● オープンソースの Web サーバーソフトウェアである。 ● PHR アプリに情報提供する HTTP サーバーである。
SL006	Apache Tomcat	Apache Tomcat 11.0.8	● Java で開発された Web アプリケーションを実行するための環境である。 ● PHR システムで利用されるWeb アプリケーションの実行環境である。
SL007	Hope LifeMark-HX	Hope LifeMark-HX	● 病院情報システムのアプリケーションである。 ● 患者診療情報や診察情報、検査や処置の依頼・結果確認診察予約などを管理・操作するアプリ群を保持する。
SL008	Hope LifeMark-コンシェルジュ	Hope LifeMark-コンシェルジュ	● 患者が利用する PHR アプリである。 ● 診察予約確認や診療情報を確認･管理するアプリ群を保持する。
SL009	診察予約アプリ	診察予約アプリ	● SL007Hope LifeMark-HX に搭載されているアプリケーションである。 ● 患者の診察予約を管理する。
SL010	PHR アプリ	PHR アプリ	● SL008Hope LifeMark-コンシェルジュに搭載されているアプリケーションである。 ● 患者の診療情報を管理
SL011	デマンド交通サービス	デマンド交通サービス	● デマンド交通向けの SaaS サービスである。 ● 利用者とデマンド車両を最適にマッチングさせ、効率的

			な車両運行を実現させるアプリケーションである。
SL012	ドライバーアプリ	ドライバーアプリ	● デマンド交通向けのアプリケーションである。 ● 車両運行を行うドライバーに対して配車指示を行うアプリケーションである。
SL013	オペレータアプリ	オペレータアプリ	● デマンド交通向けのアプリケーションである。 ● デマンド交通のオペレータに対して各種操作、確認画面を提供するアプリケーションである。
SL014	ヘルスケア MaaS Gateway デマンド配車システム I/F ライブラリ	ヘルスケア MaaS Gateway デマンド配車システム I/F ライブラリ	● PHR システムとデマンドバス配車システムを連携インターフェースで上を構成するファイル群を指す。 ● 診察予約時刻、離院予測時刻を基に作成されたデマンド交通の配車予約情報を連携する。

ソフトウェア・ライブラリの詳細を以下に示す。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】Node.js

- ベンダー
  - オープンソース
- 公式サイト
  - https://nodejs.org/ja
- 本ソフトウェア・ライブラリの概要
  - Node.js®はクロスプラットフォームに対応したフリーでオープンソースの JavaScript の実行環境である。
  - 開発者にサーバー、ウェブアプリ、コマンドラインツール、スクリプトなどを開発する環境である。
- 主な機能
  - JavaScript 実行環境：JavaScript を Web ブラウザだけでなく、サーバー上で実行する。
  - ファイルシステム操作：ファイルの読み書き、ディレクトリの作成・削除など、ファイルシステムを操作可能である。

- イメージ

CJS MJS

```
const { createServer } = require('node:http');

const hostname = '127.0.0.1';
const port = 3000;

const server = createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

図 2-14　Node.js のイメージ

【SL002】Java

- ベンダー
  - オープンソース
- 公式サイト
  - https://www.java.com/ja/
- 本ソフトウェア・ライブラリの概要
  - プラットフォーム非依存性、オブジェクト指向、自動メモリ管理などの特徴を持つ、汎用性の高いプログラミング言語である。
  - Web アプリケーション、モバイルアプリケーション、エンタープライズシステムなど、様々な分野で利用されている。
- 主な機能
  - Web アプリケーション開発：Servlet/JSP、Spring Framework、Jakarta EE などの技術を使って、Web アプリケーションを開発可能である。
  - モバイルアプリケーション開発：Android SDK を使って、Android アプリを開発可能である。
  - 豊富なライブラリ：豊富なライブラリが提供されており、開発効率を向上可能である。
- イメージ
  - -

【SL003】Swift

- ベンダー
  - オープンソース
- 公式サイト

- https://swift.org/
- 本ソフトウェア・ライブラリの概要
  - iOS、macOS、watchOS、tvOS のアプリケーション開発に使用されている。
  - Objective-C の後継として設計されており、より安全で効率的なコードを書くことができるように設計されている。
- 主な機能
  - 安全性： Swift は安全性を重視しており、変数の初期化、範囲外アクセスの防止、オプショナル型による null 値の扱いなどが組み込まれている。
  - 高速性： Swift はコンパイル言語であり、最適化されたコードを生成するため、実行速度が速い
  - 強力な型推論： Swift は型推論をサポートしており、開発者が明示的に型を指定しなくても、コンパイラが適切な型を推測する。
  - 拡張性： Swift はプロトコル指向プログラミングをサポートしており、コードの再利用性と拡張性を高度化させる。
- イメージ
  - -

【SL004】PostgreSQL

- ベンダー
  - オープンソース
- 公式サイト
  - https://www.postgresql.org/
- 本ソフトウェア・ライブラリの概要
  - 強力なオープンソースのオブジェクト関係データベース管理システムである。
  - SQL 標準に準拠しており、他のデータベースシステムからの移行が比較的容易である。
  - ACID 特性（Atomicity：原子性、Consistency：一貫性、Isolation：独立性、Durability：永続性）を保証するトランザクション処理をサポートしており、データの整合性を維持する。
- 主な機能
  - トランザクション処理が必要-ステム： 医療システムや在庫管理システムなど、データの整合性が重要なシステムに利用可能である。
  - 高い信頼性：長年の実績があり、信頼性が高いデータベースシステムである。
  - 拡張性が高い：ユーザー定義関数などを追加できるため、柔軟なカスタマイズが可能である。
- イメージ
  - -

【SL005】Apache HTTP Server

- ベンダー
  - Apache Software Foundation
- 公式サイト

- https://httpd.apache.org/

- 本ソフトウェア・ライブラリの概要
  - Apache Software Foundation によって開発・提供されているオープンソースの Web サーバーソフトウェアである。
  - 世界中で広く利用されており、特に Linux 環境での Web サーバー構築において定番の選択肢である。
  - その柔軟性と拡張性の高さから、個人のブログから大規模な商用サイトまで幅広く採用されている。
- 主な機能
  - 静的コンテンツの配信：HTML、CSS、画像、JavaScript などの静的ファイルを高速に提供する。
  - モジュールによる機能拡張：必要な機能をモジュールとして追加・削除可能（例：mod_ssl、mod_rewrite、mod_proxy）である。
- イメージ

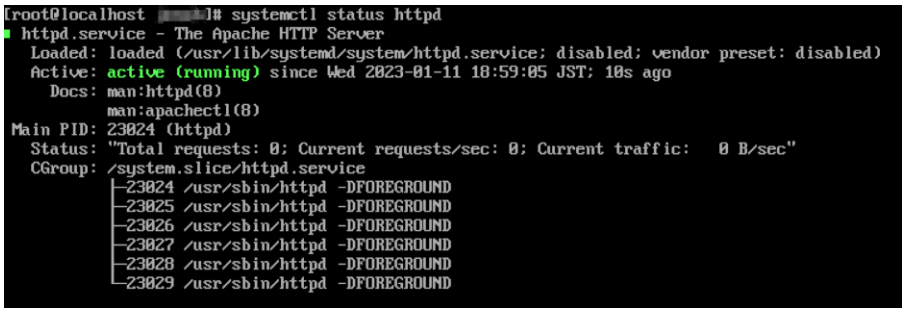


図 2-15 Apache HTTP Server のイメージ

【SL006】Apache Tomcat

- ベンダー
  - オープンソース
- 公式サイト
  - https://tomcat.apache.org
- 本ソフトウェア・ライブラリの概要
  - Tomcat は Java で書かれた Web アプリケーションを実行するための環境を提供するソフトウェアである。
  - Web ブラウザからのリクエストを受け取り、Java のプログラム（サーブレットや JSP）を実行し、その結果を Web ブラウザに返す役割である。
- 主な機能
  - Java Web アプリケーションの実行：Java サーブレット、JSP、WebSocket などを使用した Web アプリケーションを実行可能である。
  - 静的コンテンツの配信：HTML、CSS、JavaScript、画像などの静的コンテンツを配信する。
- イメージ
  - -

【SL007】Hope LifeMark-HX

- ベンダー
  - 富士通 Japan 株式会社
- 公式サイト
  - https://docs.fujitsu/documents/3-001176/hope-lifemark-hx-brochure-ja.pdf
- 本ソフトウェア・ライブラリの概要
  - 大規模・中規模病院で利用される病院情報システムである。
  - Web アプリケーションとして提供し、院内のシステムデータ統合・蓄積し、病院業務を提供する。
- 主な機能
  - 診療録管理機能：電子カルテの管理機能である。
  - 診察業務機能：患者の診察予約、検査・処置・処方などの実施計画の作成及び指示などの機能である。
  - 会計機能：患者の診療で実施された実績に基づく支払管理機能である。
- イメージ

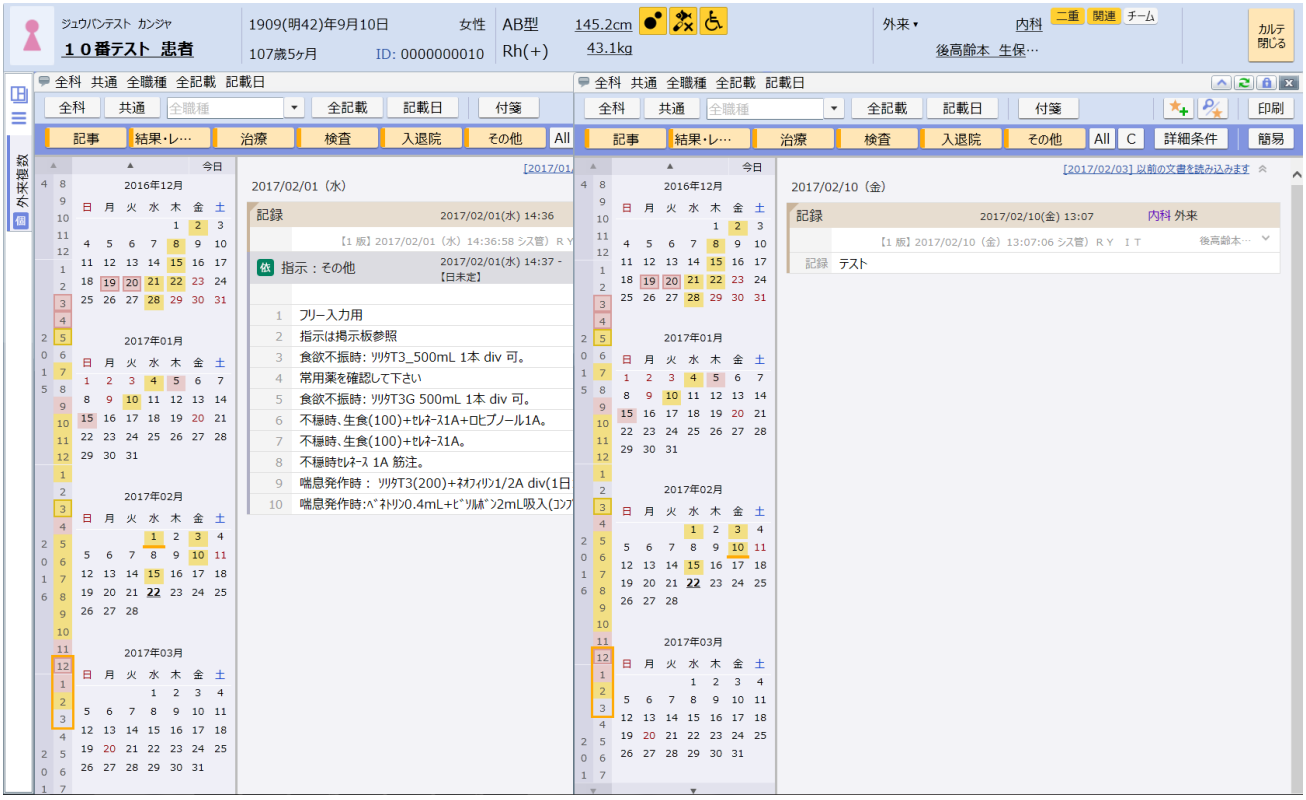


図 2-16　Hope LifeMark-HX のイメージ

【SL008】Hope LifeMark-コンシェルジュ

- ベンダー
  - 富士通 Japan 株式会社
- 公式サイト
  - https://www.fujitsu.com/jp/solutions/industry/healthcare/products/lifemarkconcierge/

- 本ソフトウェア・ライブラリの概要
  - 病院を利用する患者向けの PHR アプリである。
  - スマートフォンから予約の取得や会計の後払い機能などを提供する。
- 主な機能
  - 患者の診療情報（検査結果など）を参照する機能である。
  - 診察予約管理機能：患者の次回診察の予約取得や予約確認をする機能である。
  - 後払い管理機能：診察の会計窓口の支払いをクレジットカードで決済する機能である。
- イメージ

図 2-17　Hope LifeMark-コンシェルジュのイメージ

【SL009】診察予約アプリ

- ベンダー
  - 富士通 Japan 株式会社
- 公式サイト
  - https://docs.fujitsu/documents/3-001176/hope-lifemark-hx-brochure-ja.pdf
- 本ソフトウェア・ライブラリの概要
  - Hope LifeMark-HX に搭載された診察予約アプリである。
  - 外来患者の次回診察予約時に各病院で設定された診察枠を参照し、患者の次回予約を取得する。
- 主な機能
  - 診察予約管理機能：診察予約枠の参照、予約状況の確認、新規の診察予約の設定を行う。
- イメージ

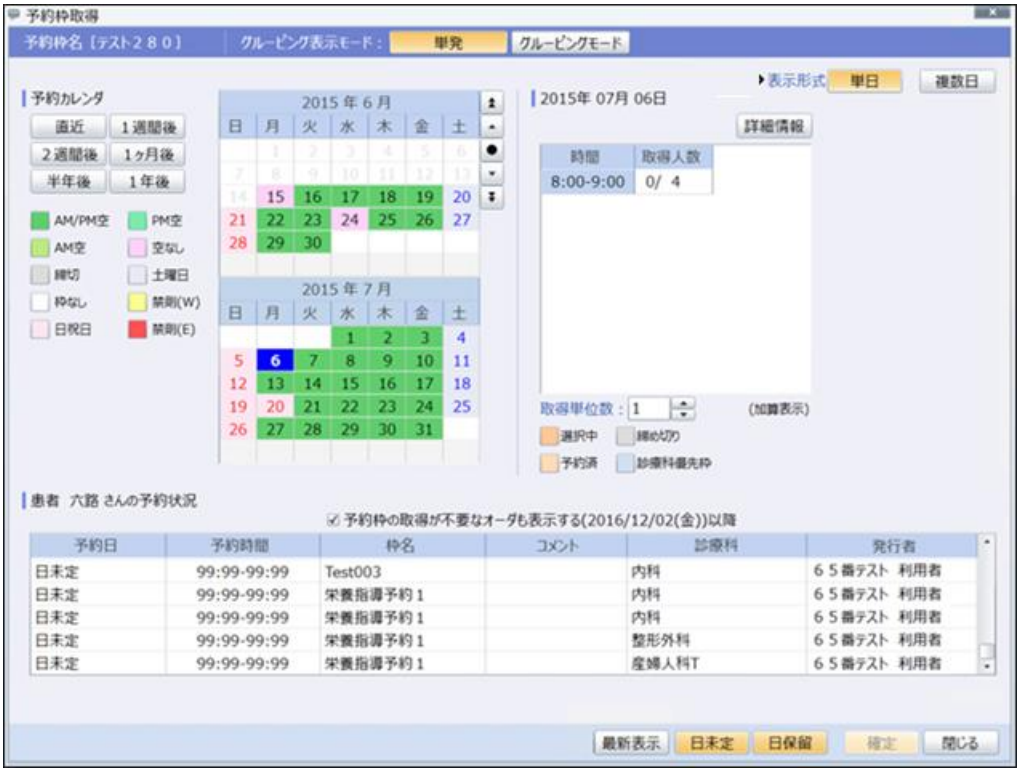


図 2-18　診察予約アプリのイメージ

【SL010】PHR アプリ

- ベンダー
  - 富士通 Japan 株式会社
- 公式サイト
  - https://www.fujitsu.com/jp/solutions/industry/healthcare/products/lifemarkconcierge/
- 本ソフトウェア・ライブラリの概要
  - Hope LifeMark-コンシェルジュに搭載された診察予約アプリである。
  - 外来患者の診療情報を参照する。
- 主な機能
  - 診療情報参照機能：患者が実施した検査の結果や受信歴などを参照する機能である。
- イメージ

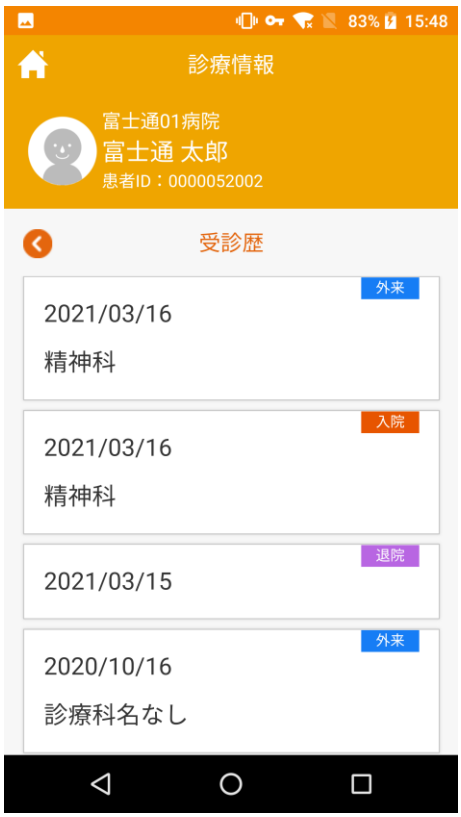


図 2-19　PHR アプリのイメージ

【SL011】デマンド交通サービス

- ベンダー
  - 富士通株式会社
- 公式サイト
  - https://www.fujitsu.com/jp/solutions/business-technology/future-mobility-accelerator/on-demand-traffic/
- 本ソフトウェア・ライブラリの概要
  - デマンド交通向けの SaaS サービスである。
- 主な機能
  - 利用者の配車予約（デマンド）を受付し、デマンドと車両を効率的にマッチングさせる。
  - 利用者のデマンドと車両のマッチング結果を基に配車指示を行う。
  - 車両の運行予定、運行実績を可視化する。
  - 運行実績データを蓄積し、必要に応じてダウンロード可能とする。
- イメージ
  - -

【SL012】ドライバーアプリ

- ベンダー
  - 富士通株式会社
- 公式サイト
  - https://www.fujitsu.com/jp/solutions/business-technology/future-mobility-accelerator/on-demand-traffic/
- 本ソフトウェア・ライブラリの概要
  - デマンド交通向けの SaaS サービスにおいて、ドライバーが利用するアプリケーションである。
- 主な機能
  - 車両運行を行うドライバーに対して配車指示を行う。
  - 利用者の乗降状態をドライバーが入力可能とし、乗降状態をデータ化する。
- イメージ

図 2-20　ドライバーアプリのイメージ

【SL013】オペレータアプリ

- ベンダー
  - 富士通株式会社
- 公式サイト
  - https://www.fujitsu.com/jp/solutions/business-technology/future-mobility-accelerator/on-demand-traffic/
- 本ソフトウェア・ライブラリの概要
  - デマンド交通向けの SaaS サービスにおいて、管理者が利用するアプリケーションである。
- 主な機能
  - デマンド交通のオペレータに対して配車予約、配車予約キャンセル機能を提供する。
  - 配車予約状況、車両の運行状況を確認できる画面を提供する。
- イメージ

図 2-21　オペレータアプリのイメージ

【SL014】ヘルスケア MaaS Gateway デマンド配車システム I/F ライブラリ

- ベンダー
  - 富士通株式会社
- 公式サイト
  - -
- 本ソフトウェア・ライブラリの概要
  - PHR システムとデマンドバス配車システムとを連携させるゲートウェイ機能である。
- 主な機能
  - 診察予約時刻を基に、デマンド交通の往路（自宅⇒病院）予約情報の生成を行い、デマンド交通の往路予約を行う。
  - 離院予測時刻を基に、デマンド交通の復路（病院⇒自宅）予約情報の生成を行い、デマンド交通の復路予約を行う。
  - 立ち寄り先と立ち寄り先の滞在時間情報を基に、デマンド交通の復路（病院⇒立ち寄り先⇒自宅）予約情報の生成を行い、デマンド交通の復路予約を行う。
  - デマンド交通の配車予約キャンセル情報を基に、該当の配車予約のキャンセルを行う。
- イメージ
  - -

2-1-5. 数理モデル・アルゴリズム（AL）の詳細

表 2-8 数理モデル・アルゴリズム一覧

※朱文字：新規開発・既存改修

ID	名称	説明	アルゴリズムを利用した機能
AL101	離院時刻予測モデル	● 外来診察における離院時刻を予測し、デマンドバスの復路配車時刻を算出するため、電子カルテシステムから取得した実績データ等を基に離院予測時刻を出力する予測モデルである。 ● 過去の診察・検査・会計等の実績データからルールベースで離院時刻予測を行う。	FN003

数理モデル・アルゴリズムの詳細を記す。なお、本業務において開発（新規・改修）を行う数理モデル・アルゴリズムを朱文字で示す。

【AL101】離院時刻予測モデル

- 本数理モデル・アルゴリズムの概要
  - 外来診察における離院時刻を予測し、デマンドバスの復路配車時刻を算出するため、電子カルテシステムから取得した実績データ等を基に離院予測時刻を出力する予測モデルである。
- 本アルゴリズムを利用した機能
  - 【FN003】離院時刻予測
- アルゴリズムの詳細
  - 電子カルテシステムから抽出した診療業務の進捗実績データ（来院時間、診察開始時間、診察終了時間、検査実施時間、会計完了時間などを記録した電子カルテシステム上のデータ）を収集し、診察当日の診療イベントに応じた在院時間の傾向を分析する。
  - 離院時刻予測モデルについては、外来時間における診察から会計までの時間の統計情報及び対象診療科で実施されている診察、検査、処置のそれぞれの統計情報を算出する。算出した統計情報から離院時刻予測を立てる利用者の当日の診察イベントに統計情報から算出した必要時間をパターンマッチングし、離院予測時刻を推定する。
  - 離院時刻予測モデルロジック

    利用者の離院予測時刻=[利用者受付時間]+[患者の平均診療時間]

    [患者の平均診療時間]=[診察時間平均値]+[検査時間平均値]+[処置時間平均値]+[会計時間平均値]

    ①[診察時間平均]：診療科・曜日・時間帯単位で計測する|[診察開始時刻]-[診察終了時刻]|
    ②[検査時間平均]：検査の実施内容単位で計測する|[検査開始時刻]-[検査終了時刻]|
    ③[処置時間平均]：処置の実施内容単位で計測する|[処置開始時刻]-[処置終了時刻]|
    ④[会計時間平均]：曜日・時間帯単位で計測する|[会計開始時刻]-[会計終了時刻]|

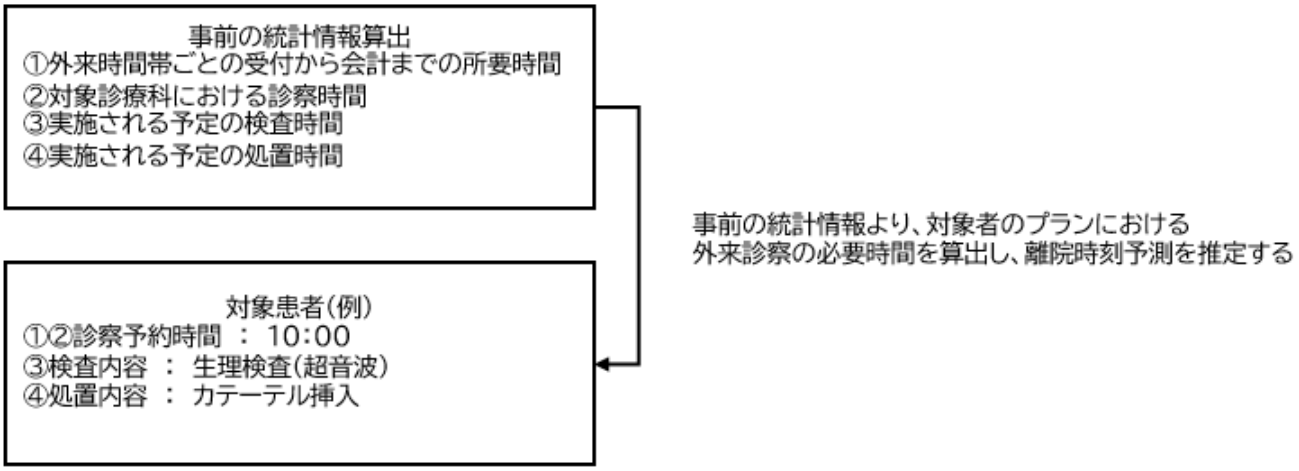


図 2-22　離院時刻予測モデルロジック

- 収集、分析するデータの想定は以下のとおり
  - ✧ 診療科・曜日・診察予約時間ごとの患者数や診療時間の変動を分析し、診察所要時間の変動を分析する。
  - ✧ 対象診療科については泌尿器科を想定とする。
  - ✧ 診察時間や実施される検査については患者ごとに内容が異なるため、泌尿器科の実績データに基づき、診察時間や検査、処置ついて実績データの統計情報を解析する。
  - ✧ 各検査の平均所要時間を収集。検査内容が在院時間に与える影響を分析する。
  - ✧ 処置の平均所要時間を収集。処置内容が在院時間に与える影響を分析する。
  - ✧ 会計待ち時間を曜日、時間帯ごとに収集する。
- 上記の収集、分析により、離院時刻の予測における重要な特徴量を選定し各特徴量と離院時刻の相関をルール化する。
- 上記で策定したルールに沿って、診察当日（当日受付チェックイン時）の特徴量に応じた離院時刻予測を行う。
  - ✧ 事前の離院時刻の予測に加え、来院当日の検査などの追加（基本的に病院側判断して対応）の有無、来院前日までの分析データの変更有無により離院時刻予測のアップデートを実施する。

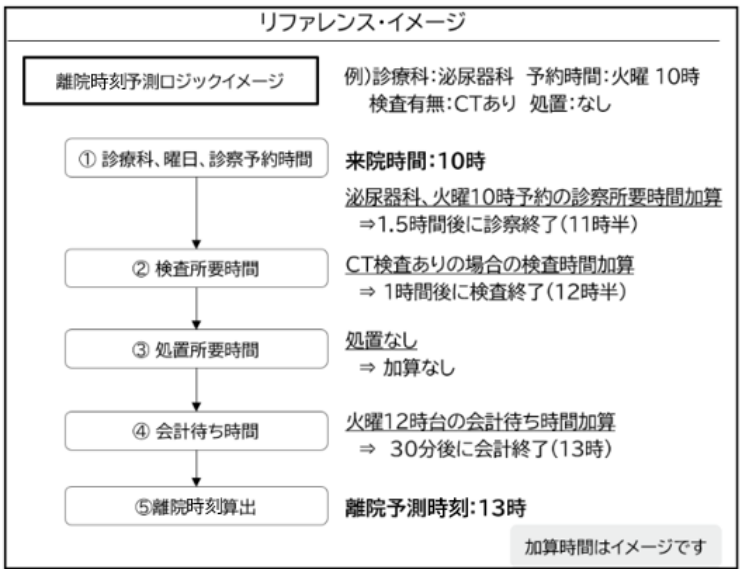


図 2-23　離院時刻予測モデルのリファレンス・イメージ

- 離院時刻予測に利用するインプットパラメータは下記を用いる。

インプットパラメータ
診療科
患者 ID
診察日
予約日時
予約枠コード
処置内容コード
処置時間
検査内容コード
検査時間
会計処理時間

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

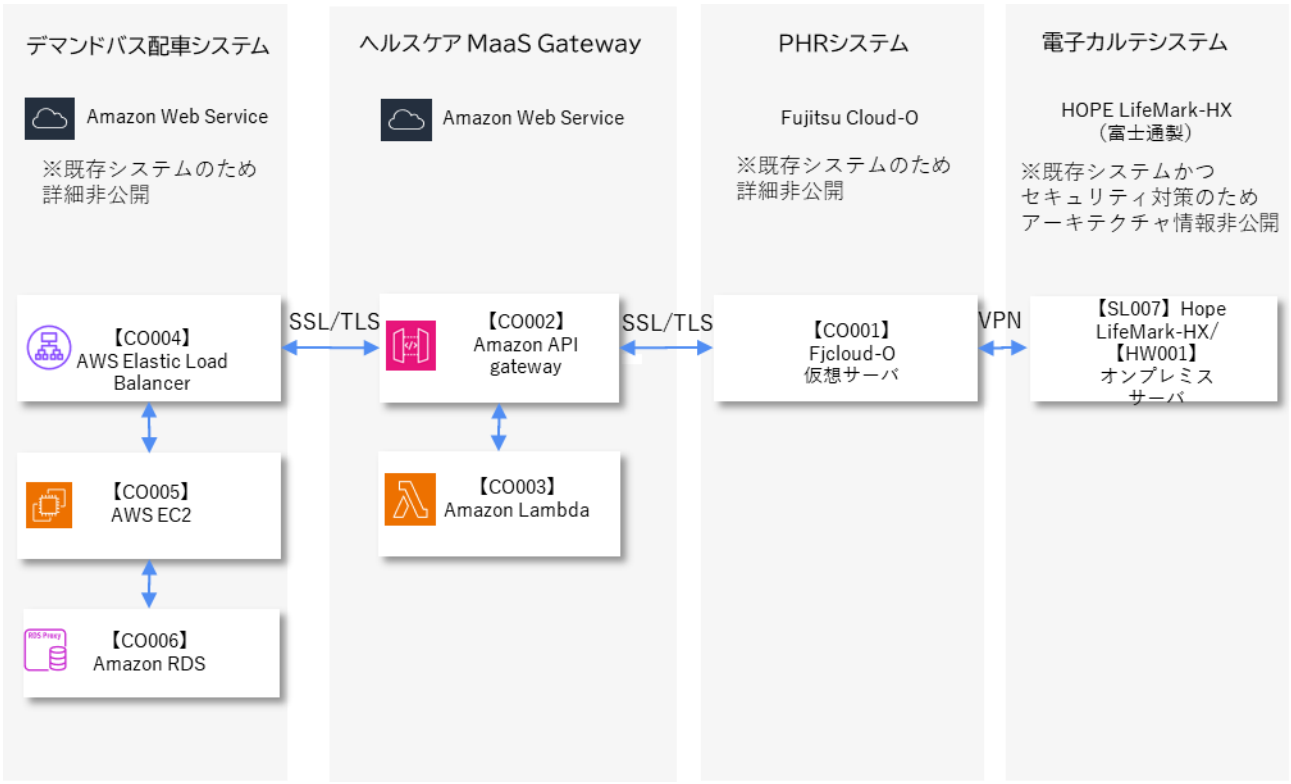


図 2-24 システムコンポーネント図

2-2-2. システムコンポーネント一覧

本節では利用するシステムコンポーネント一覧を示す。

表 2-9 システムコンポーネント一覧

※朱文字：新規開発・既存改修

ID	種別	コンポーネント名	用途
CO001	サービス	FJcloud-O	● PHR システムの業務ロジックの実行環境である。 ● 離院予測時刻を含む PHR アプリにて扱うデータを管理する。
CO002	API サーバー	Amazon API gateway	● PHR システムから配車リクエストを受け付ける API を管理する。
CO003	バックエンド処理	Amazon Lambda	● ゲートウェイのプログラムコードを実行するバックエンドの実行環境である。
CO004	ロードバランサーバー	AWS Elastic Load Balancer	● デマンドバス配車システムが外部から受け付けた処理リクエストを受付け、適切なリソースに連携する機能を持つ。
CO005	バックエンド処理	AWS EC2	● デマンドバス配車システムにおける予約や配車等の業務ロジックを実行するサーバー機能を持つ。
CO006	データベース管理	Amazon RDS	● デマンドバス配車システムにおける予約、利用者、乗降場等のデータベースを管理する機能を持つ。
CO007	サービス	電子カルテシステム	● 患者情報や診察情報、診察記録を実施・保存する機能を持つ。

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

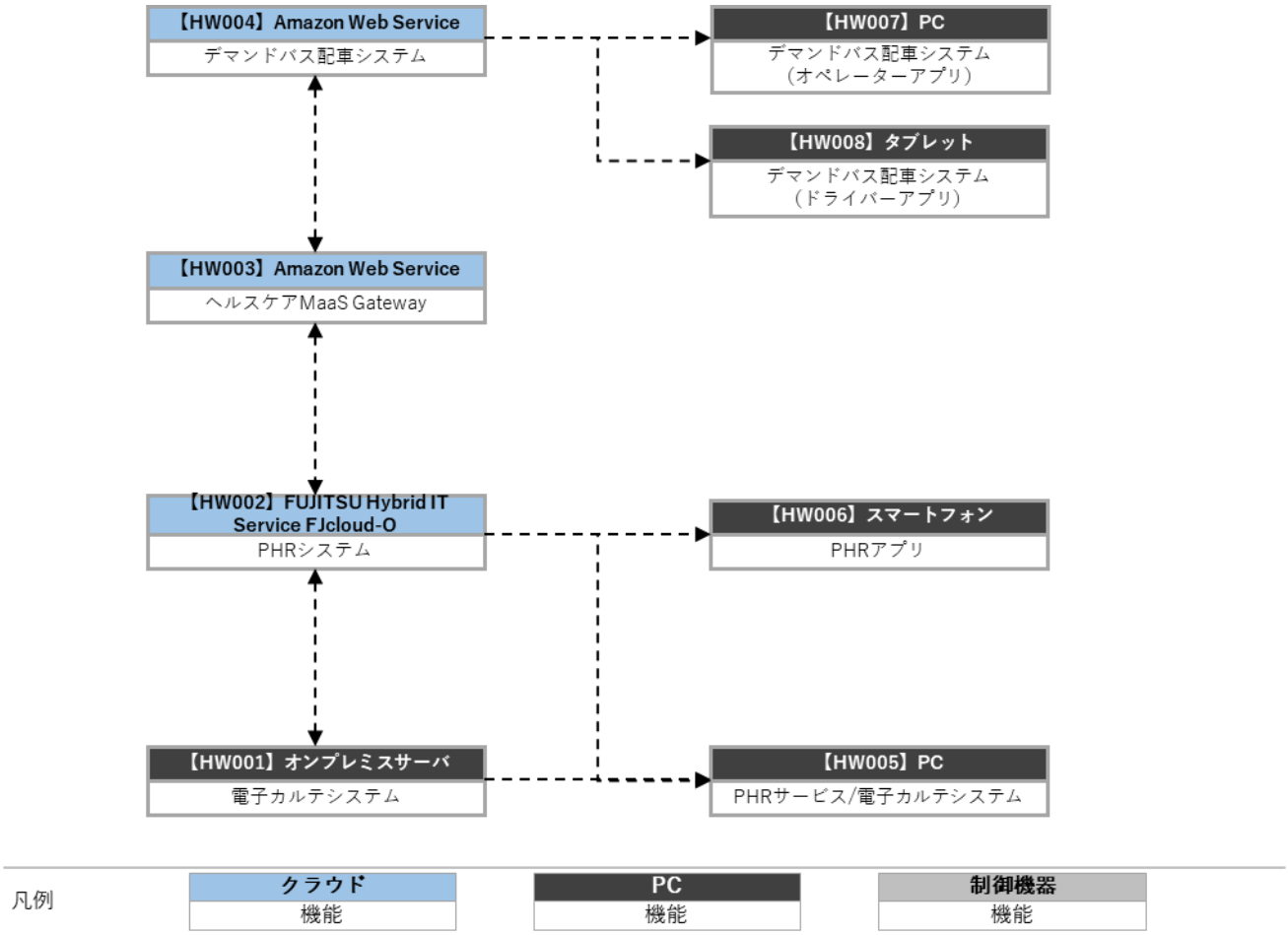


図 2-25 ハードウェアアーキテクチャ

2-3-2. ハードウェア一覧

本節では利用・開発するハードウェア一覧を示す。

表 2-10 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	SV	富士通株式会社	PRIMERGY	● 電子カルテシステムを稼働させる。
HW002	Cloud	富士通株式会社	FJcloud-O	● PHR システムを稼働させる。
HW003	Cloud	Amazon Web Services, Inc.	AWS	● ヘルスケア MaaS Gateway を稼働させる。
HW004	Cloud	Amazon Web Services, Inc.	AWS	● デマンドバス配車システムを稼働させる。
HW005	PC	FCNT	ESPRIMO	● 病院にてデマンドバス配車システムを確認する。
HW006	SP	FCNT	FCNT arrows We2 Plus M06	● 利用者にて PHR アプリを利用する。
HW007	PC	—	—	● 該当交通事業者にてデマンドバス配車システムを利用する。
HW008	TAB	Lenovo	ZAB50131JP	● 該当交通事業者にてデマンドバス配車システム（ドライバーアプリ）を利用する。

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】PRIMERGY

- 本ハードウェアの概要
  - 病院の電子カルテシステムを稼働させるサーバ
- ベンダー
  - 富士通
- 仕様・スペック
  - -
- イメージ

図 2-26 富士通　PRIMERGY

【HW002】FJcloud-O

- 本ハードウェアの概要
  - PHR システムの稼働環境である。
- ベンダー
  - 富士通株式会社
- 仕様・スペック
  - -
- イメージ
  - 公式 HP より抜粋：https://jp.fujitsu.com/solutions/cloud/fjcloud/-o/

【HW003】Amazon Web Service

- 本ハードウェアの概要
  - ヘルスケア MaaS Gateway を稼働させる。
  - 必要に応じてスペックを変更可能である。
- ベンダー
  - Amazon
- 仕様・スペック
  - メモリ：128MB
- イメージ
  - 公式 HP より抜粋：https://aws.amazon.com/jp/ec2/

【HW004】Amazon Web Service

- 本ハードウェアの概要
  - デマンドバス配車システムを稼働させる。
  - 必要に応じてスペックを変更可能である。
- ベンダー
  - Amazon
- 仕様・スペック
  - -
- イメージ
  - 公式 HP より抜粋：https://aws.amazon.com/jp/ec2/

【HW005】PC

- 本ハードウェアの概要
  - PHR システム/電子カルテシステムを利用する。
- ベンダー
  - 富士通株式会社
- 仕様・スペック

- CPU：Core（TM）i5-12500
- メモリ：8GB

- イメージ

図 2-27　ESPRIMO

【HW006】スマートフォン

- 本ハードウェアの概要
  - 利用者にて PHR アプリを利用するためのスマートデバイス。
- ベンダー
  - AndroidOS に限定する。
- 仕様・スペック
  - -
- イメージ
  - -

【HW007】PC

- 本ハードウェアの概要
  - 交通事業者にてデマンドバス配車システムを利用する。
- ベンダー
  - 各交通事業者にて利用中の端末を使用する。
- 仕様・スペック
  - 各交通事業者にて利用中の端末を使用する。

- イメージ
  - -

【HW008】タブレット

- 本ハードウェアの概要
  - ドライバーにてデマンドバス配車システムのドライバーアプリを利用
- ベンダー
  - Lenovo
- 仕様・スペック
  - OS：Android™ 12
  - プロセッサー：MediaTek Kompanio 1300T プロセッサー
  - メインメモリ：6GB
  - フラッシュメモリ：128GB
- イメージ

図 2-28　Lenovo ZAB50131JP

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

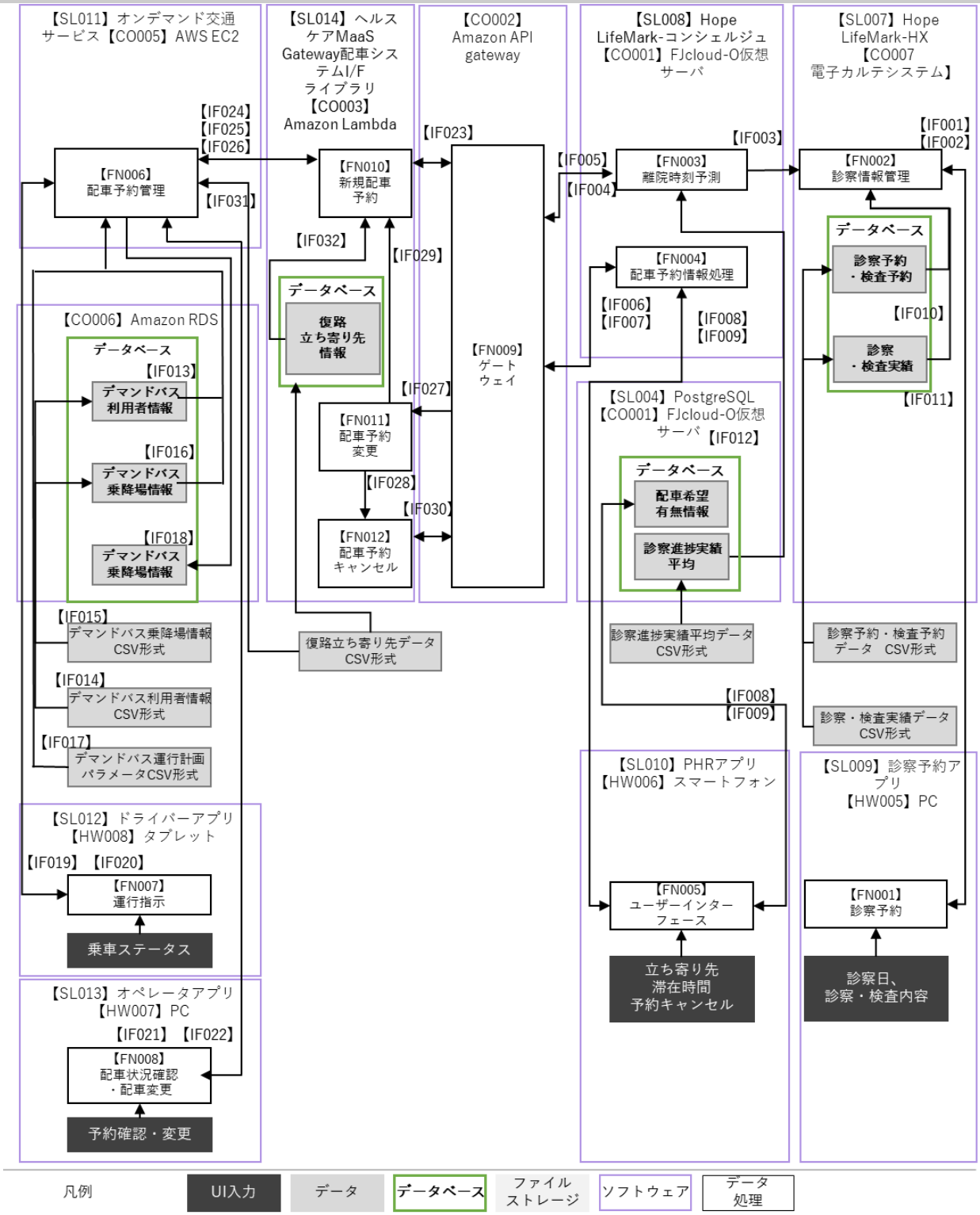


図 2-29　データアーキテクチャ

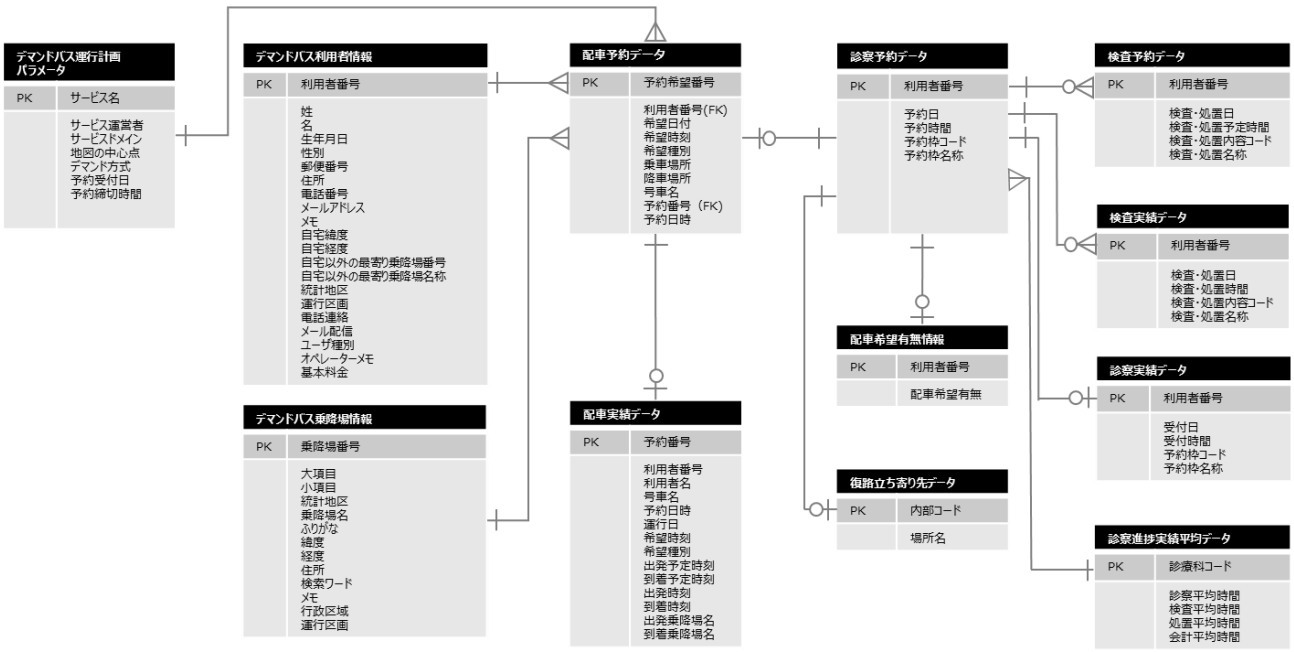

デマンドバス運行計画
パラメータ
PK
サービス名
サービス運営者
サービスドメイン
地図の中心点
デマンド方式
予約受付日
予約締切時間
デマンドバス利用者情報
PK
利用者番号
姓
名
生年月日
性別
郵便番号
住所
電話番号
メールアドレス
メモ
自宅緯度
自宅経度
自宅以外の最寄り乗降場番号
自宅以外の最寄り乗降場名称
統計地区
運行区画
電話連絡
メール配信
ユーザ種別
オペレーターメモ
基本料金
デマンドバス乗降場情報
PK
乗降場番号
大項目
小項目
統計地区
乗降場名
ふりがな
緯度
経度
住所
検索ワード
メモ
行政区域
運行区画
配車予約データ
PK
予約希望番号
利用者番号(FK)
希望日付
希望時刻
希望種別
乗車場所
降車場所
号車名
予約番号（FK)
予約日時
配車実績データ
PK
予約番号
利用者番号
利用者名
号車名
予約日時
運行日
希望時刻
希望種別
出発予定時刻
到着予定時刻
出発時刻
到着時刻
出発乗降場名
到着乗降場名
診察予約データ
PK
利用者番号
予約日
予約時間
予約枠コード
予約枠名称
配車希望有無情報
PK
利用者番号
配車希望有無
復路立ち寄り先データ
PK
内部コード
場所名
検査予約データ
PK
利用者番号
検査・処置日
検査・処置予定時間
検査・処置内容コード
検査・処置名称
検査実績データ
PK
利用者番号
検査・処置日
検査・処置時間
検査・処置内容コード
検査・処置名称
診察実績データ
PK
利用者番号
受付日
受付時間
予約枠コード
予約枠名称
診察進捗実績平均データ
PK
診療科コード
診察平均時間
検査平均時間
処置平均時間
会計平均時間

図 2-30　ER 図

2-4-2. データインターフェース一覧

本節ではデータインターフェース一覧を示す。

表 2-11 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	診療予約アプリ画面入力情報連携 IF	FN001	FN002
IF002	診療予約アプリ画面表示情報連携 IF	FN002	FN001
IF003	離院時刻予測ー診察情報連携 IF	FN002	FN003
IF004	配車予約検索確定標準 API	FN009	FN003
IF005	離院時刻予測ーゲートウェイ連携 IF	FN003	FN009
IF006	配車予約変更標準 API	FN009	FN004
IF007	配車予約キャンセル標準 API	FN009	FN004
IF008	PHR アプリ画面入力情報連携 IF	FN004	FN005
IF009	PHR アプリ画面表示情報連携 IF	FN005	FN004
IF010	診察予約・検査予約データ連携 IF	FN002	-
IF011	診察進捗実績データ IF	FN002	-
IF012	診察進捗実績平均データ IF	FN003	-
IF013	デマンドバス利用者情報連携 IF	FN006	-
IF014	デマンドバス利用者情報取込 IF	-	-
IF015	デマンドバス乗降場情報連携 IF	FN006	-
IF016	デマンドバス乗降場情報取込 IF	-	-
IF017	デマンドバス運行計画パラメータ反映 IF	FN006	-
IF018	デマンドバス運行実績出力 IF	-	FN006
IF019	ドライバー運行ステータス反映 IF	FN006	FN007
IF020	ドライバー運行指示 IF	FN007	FN006
IF021	オペレータ配車変更反映 IF	FN006	FN008
IF022	オペレータ配車情報反映 IF	FN008	FN006
IF023	ゲートウェイー新規配車予約連携 IF	FN010	FN009
IF024	予約一覧取得標準 API	FN006	FN010
IF025	デマンドバス配車予約検索標準 API	FN006	FN010
IF026	デマンドバス配車予約標準 API	FN006	FN010
IF027	ゲートウェイー配車予約変更連携 IF	FN011	FN009
IF028	配車予約変更ー配車キャンセル連携 IF	FN012	FN011
IF029	配車予約変更ー新規配車予約連携 IF	FN010	FN011

IF030	ゲートウェイー配車予約キャンセル連携 IF	FN012	FN009
IF031	デマンドバス配車予約キャンセル標準 API	FN006	FN012
IF032	立ち寄り先取得 IF	FN010	-
IF033	配車希望有無入力 IF		FN005
IF034	配車希望有無取得 IF	FN005	

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】診療予約アプリ画面入力情報連携 IF

- 本データインターフェースの概要
  - 診察予約・検査予約データを登録するためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN001】診察予約、【FN002】診療情報管理
- データ詳細

論理名	物理名	型(サイズ)
患者 ID	patient_id	String(10)
診療科	depart_id	String(3)
予約日時	reservation_date	Date(12)
予約枠コード	reservation_frame_code	String(10)
予約枠名称	reserved_mame	String(50)
予約発生日時	reserved_createDate	Date(12)

【IF002】診療予約アプリ画面表示情報連携 IF

- 本データインターフェースの概要
  - 診察予約・検査予約データを表示するためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN001】診察予約、【FN002】診療情報管理
- データ詳細

論理名	物理名	型(サイズ)
患者 ID	patient_id	String(10)
診療科	depart_id	String(3)
予約日時	reservation_date	Date(12)
予約枠コード	reservation_frame_code	String(10)
予約枠名称	reserved_name	String(50)

【IF003】離院時刻予測－診察情報連携 IF＜新規開発＞

- 本データインターフェースの概要
  - 診察進捗実績データからデータ抽出するためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN002】診察情報管理、【FN003】離院時刻予測

- データ詳細

論理名	物理名	型(サイズ)
診療科	clinical_department	String(4)
患者 ID	patient_id	String(10)
診察日	examination_date	TIMESTAMP (8)
予約日時	reservation_date	TIMESTAMP (8)
予約枠コード	reservation_frame_code	String(8)
処置内容コード	treatment_detail_code	String(8)
処置時間	treatment_time	TIMESTAMP (8)
検査内容コード	inspection_detail_code	String(8)
検査時間	inspection_time	TIMESTAMP (8)
会計処理時間	account_time	TIMESTAMP (8)

【IF004】配車予約検索確定標準 API ＜新規開発＞

- 本データインターフェースの概要
  - 病院到着希望日時に基づくデマンドバスの往路（自宅→病院）と病院出発希望日時に基づく復路（病院→自宅）の予約を行う。また、予約確定時には PHR アプリへ予約内容の通知を行う。
- 本インターフェースを利用する機能
  - 【FN003】離院時刻予測、【FN009】ゲートウェイ
- 共通仕様
  - プロトコル
    - HTTPS
  - メソッド
    - POST
  - パス
    - /reservations
  - 共通リクエストパラメータ

項目	名称	説明	値	必須
利用者番号	user_id	利用者番号	123456：String	○
到着希望時刻	arrival_demand_time	病院到着希望日時	Time：ISO8601 形式	○-
出発希望時刻	departure_demand_time	病院出発希望日時	Time：ISO8601 形式	○

  - 共通レスポンスパラメータ

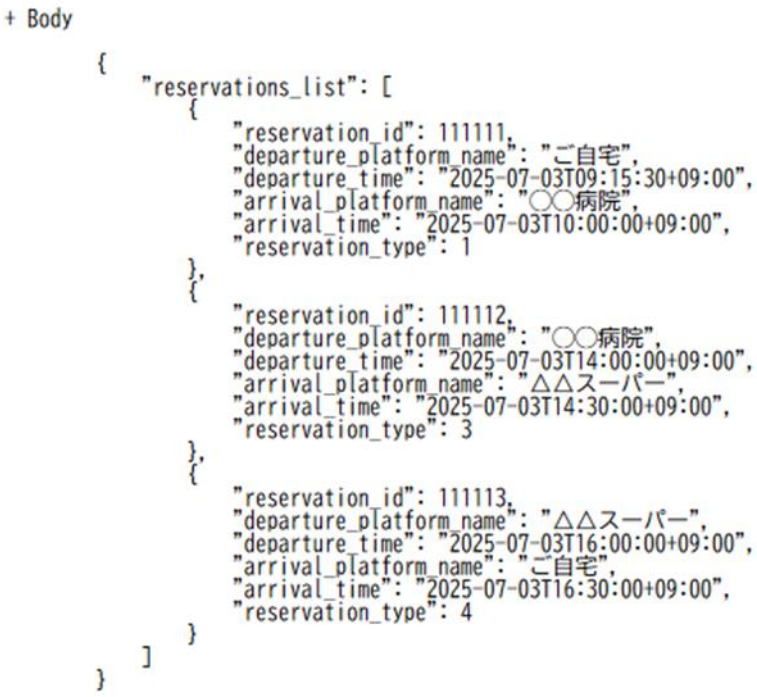
項目	名称	説明	値
予約リスト	reservations_list	予約リスト（自宅⇒病院、病院⇒自宅の 2 件の予約 Object を返却）	予約が成立した場合、予約 Object を返却

-	{	Object 型　多重度 2	-
配車予約 ID	reservation_id	配車予約 ID	123456：Number
出発地	departure_platform_name	出発地（自宅又は病院）	○○様自宅：String
出発時刻	departure_time	出発時刻	Time：ISO8601 形式
到着地	arrival_platform_name	到着地（病院又は自宅）	○○病院：String
到着時刻	arrival_time	到着時刻	Time：ISO8601 形式
予約種別	reservation_type	予約種別。以下のルールで返却する。 1: 行きの予約（例：自宅→病院） 2. 帰りの予約かつ、立ち寄り先のない病院→自宅の予約（例: 病院→自宅） 3. 帰りの予約かつ、病院→立ち寄り先の予約（例：病院→スーパー） 4. 帰りの予約かつ、立ち寄り先→自宅までの予約（例: スーパー→自宅）	1：行きの予約 2：帰りの予約かつ、立ち寄り先のない病院→自宅の予約 3：帰りの予約かつ、病院→立ち寄り先の予約 4：帰りの予約かつ、立ち寄り先→自宅までの予約
-	}	-	-

- リクエストサンプル

```
+ Body
        {
            "user_id": "123456",
            "arrival_demand_time": "2025-07-03T10:15:30+09:00",
            "departure_demand_time": "2025-07-03T15:15:30+09:00",
            "way_points_id": 1,
            "stay_duration_minutes": 30
        }
```

- レスポンスサンプル



```
+ Body
{
    "reservations_list": [
        {
            "reservation_id": 111111,
            "departure_platform_name": "ご自宅",
            "departure_time": "2025-07-03T09:15:30+09:00",
            "arrival_platform_name": "○○病院",
            "arrival_time": "2025-07-03T10:00:00+09:00",
            "reservation_type": 1
        },
        {
            "reservation_id": 111112,
            "departure_platform_name": "○○病院",
            "departure_time": "2025-07-03T14:00:00+09:00",
            "arrival_platform_name": "△△スーパー",
            "arrival_time": "2025-07-03T14:30:00+09:00",
            "reservation_type": 3
        },
        {
            "reservation_id": 111113,
            "departure_platform_name": "△△スーパー",
            "departure_time": "2025-07-03T16:00:00+09:00",
            "arrival_platform_name": "ご自宅",
            "arrival_time": "2025-07-03T16:30:00+09:00",
            "reservation_type": 4
        }
    ]
}
```

【IF005】離院時刻予測ーゲートウェイ連携 IF　＜新規開発＞

- 本データインターフェースの概要
  - 離院時刻をゲートウェイ機能に連携した後の処理正常終了を確認する。レスポンス自体は FN004 配車予約情報処理に引き継がれる。
- 本インターフェースを利用する機能
  - 【FN003】離院時刻予測、【FN009】ゲートウェイ
- データ詳細

名称	説明	値	必須
user_id	利用者番号	String	○
arrival_demand_time	病院到着希望日時	Time ISO8601 形式	○
departure_demand_time	病院出発希望日時	Time ISO8601 形式	○

【IF006】配車予約変更標準 API　＜新規開発＞

- 本データインターフェースの概要
  - 離院時刻が変更になった場合、又は利用者の希望により往路（自宅→病院）、復路（病院→自宅）又はその両方の予約を変更する。また、立ち寄り先が選択された場合は、復路（病院→自宅）の予約を病院→立ち寄り先→自宅に変更する。変更予約確定時には PHR アプリへ変更内容の通知を行う。
- 本インターフェースを利用する機能
  - 【FN004】配車予約情報処理、【FN009】ゲートウェイ

- 共通仕様
  - プロトコル
    - HTTPS
  - メソッド
    - POST
  - パス
    - /reservations
  - 共通リクエストパラメータ

項目	名称	説明	値	必須
利用者番号	user_id	利用者番号	123456：String	○
予約 ID	reservation_ids	変更対象の予約 ID（Array 型）	123456：Number	○
到着希望時刻	arrival_demand_time	病院到着希望日時	Time：ISO8601 形式	○
出発希望時刻	departure_demand_time	病院出発希望日時	Time：ISO8601 形式	○
立寄り先 ID	way_points_id	立ち寄り先 ID	12：Number	―
滞在時間	stay_duration_minutes	立ち寄り先の滞在時間	30：Number	―

- 共通レスポンスパラメータ

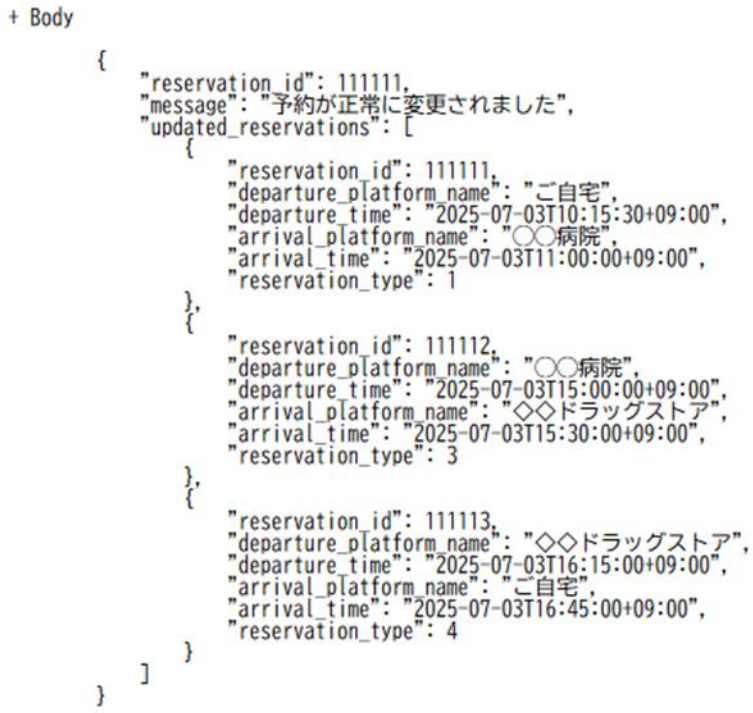
項目	名称	説明	値
予約リスト	reservations_list	予約リスト（自宅⇒病院の予約、病院⇒自宅の予約、又は病院⇒立ち寄り先⇒自宅の最大 3 件の予約 Object を返却）	予約が成立した場合、予約 Object を返却
—	{	Object 型　多重度最大 3	—
配車予約 ID	reservation_id	配車予約 ID	123456：Number
出発地	departure_platform_name	出発地（自宅、病院又は立ち寄り先）	〇〇様自宅：String
出発時刻	departure_time	出発時刻	Time：ISO8601 形式
到着地	arrival_platform_name	到着地（病院、立ち寄り先又は自宅）	〇〇病院：String
到着時刻	arrival_time	到着時刻	Time：ISO8601 形式
予約種別	reservation_type	予約種別。以下のルールで返却する。 1: 行きの予約（例：自宅→病院） 2. 帰りの予約かつ、立ち寄り先のない病院→自宅の予約（例: 病院→自宅） 3. 帰りの予約かつ、病院→立ち寄り先の予約（例：病院→スーパー） 4. 帰りの予約かつ、立ち寄り先→自宅までの予約（例: スーパー→自宅）	1：行きの予約 2：帰りの予約かつ、立ち寄り先のない病院→自宅の予約 3：帰りの予約かつ、病院→立ち寄り先の予約 4：帰りの予約かつ、立ち寄り先→自宅までの予約
—	}	—	—

- リクエストサンプル

+ Body


```
{
    "user_id": "123456",
    "reservation_id": 111111,
    "arrival_demand_time": "2025-07-03T11:15:30+09:00",
    "departure_demand_time": "2025-07-03T16:15:30+09:00",
    "way_points_id": 2,
    "stay_duration_minutes": 45
}
```

- レスポンスサンプル



+ Body


```
{
    "reservation_id": 111111,
    "message": "予約が正常に変更されました",
    "updated_reservations": [
        {
            "reservation_id": 111111,
            "departure_platform_name": "ご自宅",
            "departure_time": "2025-07-03T10:15:30+09:00",
            "arrival_platform_name": "○○病院",
            "arrival_time": "2025-07-03T11:00:00+09:00",
            "reservation_type": 1
        },
        {
            "reservation_id": 111112,
            "departure_platform_name": "○○病院",
            "departure_time": "2025-07-03T15:00:00+09:00",
            "arrival_platform_name": "◇◇ドラッグストア",
            "arrival_time": "2025-07-03T15:30:00+09:00",
            "reservation_type": 3
        },
        {
            "reservation_id": 111113,
            "departure_platform_name": "◇◇ドラッグストア",
            "departure_time": "2025-07-03T16:15:00+09:00",
            "arrival_platform_name": "ご自宅",
            "arrival_time": "2025-07-03T16:45:00+09:00",
            "reservation_type": 4
        }
    ]
}
```


【IF007】配車予約キャンセル標準 API ＜新規開発＞

- 本データインターフェースの概要
  - PHR アプリ上で利用者によりデマンドバスの配車予約がキャンセルされた場合、それにひも付く配車予約をキャンセル処理する。キャンセル処理確定時には PHR アプリへ確定の通知を行う。
- 本インターフェースを利用する機能
  - 【FN004】配車予約情報処理、【FN009】ゲートウェイ
- 共通仕様
  - プロトコル
    - HTTPS
  - メソッド
    - DELETE
  - パス
    - /reservations
  - 共通リクエストパラメータ

項目	名称	説明	値	必須
利用者番号	user_id	利用者番号	123456：String	○
予約 ID	reservation_ids	変更対象の予約 ID（Array 型）、最大 3 個	123456：Number	○

➢　リクエストサンプル

+ Body

```
{
    "user_id": "000001",
    "reservation_ids": [111111, 111112, 111113]
}
```

【IF008】PHR アプリ画面入力情報連携 IF＜新規開発＞

- 本データインターフェースの概要
- 配車予約入力情報を PHR システムへ連携するためのインターフェースである。
  - ➢　【FN004】配車予約情報処理、【FN005】ユーザーインターフェース
- データ詳細

論理名	物理名	型(サイズ)
希望到着時刻	desired_arrival_time	ISO8601 形式 hh:mm
希望出発時刻	desired_ departure_time	ISO8601 形式 hh:mm
立ち寄り先コード	stopover_points_code	String (10)
滞在時間コード	staying_time_code	String (10)

【IF009】PHR アプリ画面表示情報連携 IF＜新規開発＞

- 本データインターフェースの概要
  - ➢　配車予約結果情報を PHR アプリに表示するためのインターフェースである。
- 本インターフェースを利用する機能
  - ➢　【FN004】配車予約情報処理、【FN005】ユーザーインターフェース
- データ詳細

論理名	物理名	型(サイズ)
配車予約 ID	reservation_id	String(10)
自宅出発時刻	outward_departure_time	ISO8601 形式 hh:mm
到着時刻	arrival_time	ISO8601 形式 hh:mm
希望到着時刻	desired_ arrival_time	ISO8601 形式 hh:mm
病院出発時刻	return_departure_time	ISO8601 形式 hh:mm
希望出発時刻	desired_return_departure_time	ISO8601 形式 hh:mm

立ち寄り先コード	stopover_points_code	String (10)
滞在時間コード	staying_time_code	String (10)
離院予測時刻	estimated_finish_time	ISO8601 形式
診察時間	clinical_time	ISO8601 形式 yyyy/mm/dd hh:mm:ss

【IF010】診察予約・検査予約データ連携 IF

- 本データインターフェースの概要
  - 診察予約・検査予約データをデータベースへ登録するためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN002】診療情報管理
- データ詳細

論理名	物理名	型(サイズ)
患者 ID	patient_id	String(10)
診療科	depart_id	String(3)
予約日時	reservation_date	Date(12)
予約枠コード	reservation_frame_code	String(10)
予約枠名称	reserved_mame	String(50)
予約発生日時	reserved_createDate	Date(12)

【IF011】診察進捗実績データ IF

- 本データインターフェースの概要
  - 診察進捗実績データをデータベースへ登録するためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN002】診療情報管理
- データ詳細

論理名	物理名	型(サイズ)
診療科	clinical_department	String(4)
患者 ID	patient_id	String(10)
予約日時	reservation_date	TIMESTAMP (8)
予約枠コード	reservation_frame_code	String(8)
検査内容コード	inspection_detail_code	String(8)

【IF012】診察進捗実績平均データ IF＜新規開発＞

- 本データインターフェースの概要
  - 診察進捗実績平均データをデータベースへ登録するためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN003】離院時刻予測
- データ詳細

論理名	物理名	型(サイズ)
診療科	clinical_department	String(4)
予約日時	reservation_date	TIMESTAMP (8)
予約枠コード	reservation_frame_code	String(8)
検査内容コード	inspection_detail_code	String(8)
診察平均時間	examination_averagetime	Date(12)
検査平均時間	inspection_averagetime	Date(12)
会計平均時間	payment_averagetime	Date(12)

【IF013】デマンドバス利用者情報連携 IF

- 本データインターフェースの概要
  - 車両予約や配車時にデマンドバス利用者情報を参照するためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN009】配車予約管理
- データ詳細

論理名	物理名	型(サイズ)
利用者番号	login_id	String (6)
姓	last_name	String (6)
名	first_name	String (6)
電話番号	telephone_number	String (11)
メールアドレス	email	String (50)

【IF014】デマンドバス利用者情報取込 IF

- 本データインターフェースの概要
  - デマンドバス利用者情報をデマンドバス配車システムに取り込むためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN009】配車予約管理
- データ詳細

論理名	物理名	型(サイズ)
利用者番号	login_id	String (6)
姓	last_name	String (6)
名	first_name	String (6)
電話番号	telephone_number	String (11)
メールアドレス	email	String (50)

【IF015】デマンドバス乗降場情報連携 IF

- 本データインターフェースの概要
  - 車両予約や配車時にデマンドバス乗降場情報を参照するためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN009】配車予約管理
- データ詳細

論理名	物理名	型(サイズ)
大項目	large_category_name	String (50)
小項目	small_category_name	String (10)
識別子	id	Int (3)
名称	name	String (10)
緯度	latitude	float (15)
経度	longitude	float (15)

【IF016】デマンドバス乗降場情報取込 IF

- 本データインターフェースの概要
  - デマンドバス乗降場情報をデマンドバス配車システムに取り込むためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN009】配車予約管理
- データ詳細

論理名	物理名	型(サイズ)
大項目	large_category_name	String (50)
小項目	small_category_name	String (10)
識別子	id	Int (3)
名称	name	String (10)
緯度	Latitude	float (15)
経度	longitude	float (15)

【IF017】デマンドバス運行計画パラメータ反映 IF

- 本データインターフェースの概要
  - デマンドバス運行計画パラメータをデマンドバス配車システムに取り込むためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN009】配車予約管理
- データ詳細
  - -

【IF018】デマンドバス運行実績出力 IF

- 本データインターフェースの概要
  - デマンドバスの車両実績を出力するためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN009】配車予約管理
- データ詳細

論理名	物理名	型(サイズ)
利用者番号	login_id	String (6)
予約番号	reservation_id	Int (6)
号車名	vehicle_name	Text(10)
運行日	date	DATE (8)
出発時刻	latest_departure_time	TIMESTAMP (8)
到着時刻	latest_arrival_time	TIMESTAMP (8)

【IF019】ドライバー運行ステータス反映 IF

- 本データインターフェースの概要
  - ドライバーアプリ（タブレット）にて操作された乗車ステータスの情報を配車予約データに連携するインターフェースである。
- 本インターフェースを利用する機能
  - 【FN010】運行指示
- データ詳細
  - -

【IF020】ドライバー運行指示 IF

- 本データインターフェースの概要
  - 配車予約結果をドライバーアプリ（タブレット）に連携するインターフェースである。
- 本インターフェースを利用する機能
  - 【FN010】運行指示
- データ詳細
  - -

【IF021】オペレータ配車変更反映 IF

- 本データインターフェースの概要
  - 配車予約結果をオペレータアプリ（PC）に連携するインターフェースである。
    （本実証では使用しない）
- 本インターフェースを利用する機能
  - 【FN011】配車状況確認・配車変更
- データ詳細
  - -

【IF022】オペレータ配車情報反映 IF

- 本データインターフェースの概要
  - オペレータアプリ（PC）にて操作された予約変更の情報を配車予約データに反映するインターフェースである。
- 本インターフェースを利用する機能
  - 【FN011】配車状況確認・配車変更
- データ詳細
  - -

【IF023】ゲートウェイー新規配車予約連携 IF　＜新規開発＞

- 本データインターフェースの概要
  - 配車予約検索確定標準 API にて連携された内容を新規配車予約機能に連携し結果を呼出し元に返却する。
- 本インターフェースを利用する機能
  - 【FN009】ゲートウェイ、【FN010】新規配車予約
- データ詳細
  - 入力

名称	説明	値	必須
user_id	利用者番号	String	○
arrival_demand_time	病院到着希望日時	Time ISO8601 形式	○-
departure_demand_time	病院出発希望日時	Time ISO8601 形式	○

  - 出力

名称	説明
reservations_list	予約リスト（自宅⇒病院、病院⇒自宅の 2 件の予約 Object を返却）
{	Object 型　多重度 2
reservation_id	配車予約 ID
departure_platform_name	出発地（自宅又は病院）
departure_time	出発時刻
arrival_platform_name	到着地（病院又は自宅）
arrival_time	到着時刻
}	

【IF024】予約一覧取得標準 API

- 本データインターフェースの概要
  - デマンドバスの利用者が予約した予約一覧を取得するためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN006】配車予約管理、【FN012】配車予約キャンセル
- 利用するデータインターフェース
  - 以下のデマンドバスシステム標準 API を利用する。

ユーザーの予約一覧を取得

パスパラメータで指定したユーザーが登録した予約情報を、クエリパラメータの指定に応じて複数の条件で予約情報を絞り込んで、複数件取得します。
一覧表示や履歴画面など、複数の予約情報を取得する用途で利用されます。
各予約オブジェクトのレスポンス構造は `GET /reservations/{id}` と共通です。
各条件は組み合わせて使用可能で、AND 条件として評価されます。

主な検索条件とその動作は以下のとおりです：

- `pickup_date_from`
  乗車予定日のフィルター開始日を指定します（この日を含みます）。
  指定がない場合は制限されません。
- `pickup_date_to`
  乗車予定日のフィルター終了日を指定します（この日を含みます）。
  指定がない場合は制限されません。
- `reservation_ids`
  明示的に指定された予約IDのリストに含まれる予約を対象とします（完全一致、カンマ区切り）。
- `status`
  指定された予約ステータス（`tentative`, `confirmed`, `no_show`, `in_transit`, `completed`, `cancelled`）に一致する予約を対象とします。
  複数指定可能（OR 条件）です。

リクエスト例:

```
GET /passengers/{id}/reservations?reservation_ids=1,2,3&status=tentative,confirmed
```

PATH PARAMETERS

id required	string Example: `ps012345678` 予約を登録したユーザーIDを指定します。

pickup_date_from

string <date>

Example: `pickup_date_from=2025-07-01`

乗車予定日のフィルター開始日です（この日を含みます）。形式はRFC3339のdate形式文字列（例：2025-06-19）です。

指定がない場合は制限されません。

---

pickup_date_to

string <date>

Example: `pickup_date_to=2025-07-05`

乗車予定日のフィルター終了日です（この日を含みます）。形式はRFC3339のdate形式文字列（例：2025-06-19）です。

指定がない場合は制限されません。

---

reservation_ids

Array of strings

Example: `reservation_ids=1,2,3`

取得対象の予約IDをカンマ区切りで指定します。

---

status

Array of strings

Items Enum: `"tentative"` `"confirmed"` `"no_show"` `"in_transit"` `"completed"` `"cancelled"`

Example: `status=tentative,confirmed`

フィルター対象とする予約のステータスを複数指定します。

指定がない場合はすべてのステータスを含む結果が返されます。

- `tentative`
  仮予約。乗車便が未確定で、締切前の状態です。確定処理が行われるまではこの状態となります。
- `confirmed`
  確定予約。 確定予約。乗車便が成立した状態です。
- `in_transit`
  乗車中。車両が運行中で、乗客が乗車している状態です。
- `completed`
  降車済み。乗車が完了し、正常に終了した予約です。
- `no_show`
  未乗車。予約は確定していたものの、乗客が現れず乗車しなかった状態です。
- `cancelled`
  キャンセル済み。利用者またはシステムにより予約が取り消された状態です。（仮予約・確定予約いずれからも遷移します）

offset
integer >= 0
Default: 0
Example: offset=30
取得する結果の開始位置を指定します。

limit
integer [1 .. 100]
Default: 20
Example: limit=10
取得する結果の最大件数を指定します。

Responses

200 OK

RESPONSE SCHEMA: application/json

reservations
required
Array of objects >= 0 items
予約情報のリストです。

Array (>= 0 items) [

id
required
string
登録済み予約の一意な識別子です。

passenger_id
required
string
予約を登録したユーザーの一意な識別子です。

status
required

string
Enum: `"tentative"` `"confirmed"` `"no_show"` `"in_transit"` `"completed"` `"cancelled"`
予約のステータスです。

- `tentative`
  仮予約。乗車便が未確定で、締切前の状態です。確定処理が行われるまではこの状態となります。
- `confirmed`
  確定予約。 確定予約。乗車便が成立した状態です。
- `in_transit`
  乗車中。車両が運行中で、乗客が乗車している状態です。
- `completed`
  降車済み。乗車が完了し、正常に終了した予約です。
- `no_show`
  未乗車。予約は確定していたものの、乗客が現れず乗車しなかった状態です。
- `cancelled`
  キャンセル済み。利用者またはシステムにより予約が取り消された状態です。（仮予約・確定予約いずれからも遷移します）

---

service_id
required

string
予約便候補が属するサービスの一意な識別子です。

---

pickup ›
required

pickup

object or object or object
乗車場所と予定時刻の情報です。

One of **object** | object | object

- **type** (required)
  string
  Value: `"fixed_stop"`
  乗降場所の種類を示します。`fixed_stop` の場合、固定の乗降場所を示します。

- **stop_id** (required)
  string
  固定の乗降場所のIDです。`type` が `fixed_stop` の場合のみ設定されます。

- **display_name** (required)
  string
  乗降場所の表示名です。`type` が `fixed_stop` の場合、乗降場所の名称が設定されます。

- **location** (required)
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

  - **type** (required)
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）

  - **coordinates** (required)
    Array of numbers <double> = 2 items [items <double >]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

- **datetime** (required)
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of　object　object　object

- **type** (required)
  string
  Value: `"custom_location"`
  乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

- **display_name** (required)
  string
  乗降場所の表示名です。`type` が `custom_location` の場合「任意地点」が設定されます。

- **location** (required)
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

  - **type** (required)
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）

  - **coordinates** (required)
    Array of numbers <double>　= 2 items　[items <double >]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

- **datetime** (required)
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of　object　object　**object**

- **type** *required*
  string
  Value: `"home"`
  乗降場所の種類を示します。`home` の場合、自宅を示します。
- **passenger_id** *required*
  string
  自宅の位置を指定する場合のユーザーIDです。`type` が `home` の場合のみ設定されます。
- **display_name** *required*
  string
  乗降場所の表示名です。`type` が `home` の場合「自宅」が設定されます。
- **location** *required*
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。
  - **type** *required*
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）
  - **coordinates** *required*
    Array of numbers <double>　= 2 items　[items <double >]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。
- **datetime** *required*
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

dropoff
required

object or object or object
降車場所と予定時刻の情報です。

One of object | object | object

type
required

string
Value: `"fixed_stop"`
乗降場所の種類を示します。`fixed_stop` の場合、固定の乗降場所を示します。

stop_id
required

string
固定の乗降場所のIDです。`type` が `fixed_stop` の場合のみ設定されます。

display_name
required

string
乗降場所の表示名です。`type` が `fixed_stop` の場合、乗降場所の名称が設定されます。

location
required

object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required

string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required

Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required

string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

dropoff
required

object or object or object
降車場所と予定時刻の情報です。

One of | object | object | object

type
required

string
Value: `"custom_location"`
乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

display_name
required

string
乗降場所の表示名です。`type` が `custom_location` の場合「任意地点」が設定されます。

location
required

object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required

string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required

Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required

string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

dropoff
required

object or object or object
降車場所と予定時刻の情報です。

One of object object object

type
required

string
Value: `"home"`
乗降場所の種類を示します。`home` の場合、自宅を示します。

passenger_id
required

string
自宅の位置を指定する場合のユーザーIDです。`type` が `home` の場合のみ設定されます。

display_name
required

string
乗降場所の表示名です。`type` が `home` の場合「自宅」が設定されます。

location
required

object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required

string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required

Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required

string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

passenger_count
required

Array of objects　non-empty
乗客の種別ごとの人数を示すリストです。

Array (non-empty) [

- passenger_type_id
  required
  string
  乗客の種別です。

- count
  required
  integer <int32>　>= 1
  この種別の乗客数です。

]

accessibility_feature_count
required

Array of objects　>= 0 items
利用する乗客補助機能の種類と数です。

Array (>= 0 items) [

- type
  required
  string
  乗客補助機能の種類です。利用可能な種類はサービスの詳細情報（GET /v1/services/{id}）で取得できる `supported_accessibility_features` を参照してください。

- count
  required
  integer <int32>　>= 1
  この種類の乗客補助機能の必要数です。

]

- **fare** (required) — object
  運賃の情報です。
  - **per_passenger_type** (required) — Array of objects, >= 0 items
    乗客の種別ごとの運賃小計を示すリストです。

    Array (>= 0 items) [
    - **passenger_type_id** (required) — string
      乗客の種別です。
    - **subtotal** (required) — integer <int32>, >= 0
      この種別の運賃小計（円）です。

    ]
  - **total** — integer <int32>, >= 0
    運賃の合計（円）です。
- **route_distance** — integer <int32>
  乗車地点から降車地点までのルート探索に基づく推定距離（メートル）です。
  直線距離ではなく、道路網上の最短ルートを基に計算された値です。

vehicle ⌄ object
この予約で利用する車両の情報です。
vehicle に値があるときは `id` と `name` は必ず値が設定されます。

- id — string
  車両IDです。
- name — string
  車両名です。
- number_plate — string
  車両のナンバープレート番号です。
- capacity ⌄ object
  車両の座席数と乗客補助機能の情報です。
  各車両は、最大座席数と利用可能な乗客補助機能の種類とその最大数を持ちます。
  - seats — integer <int32> >= 1
    車両の最大座席数です。
  - accessibility_features › Array of objects
    車両に備わっている乗客補助機能の種類とそれぞれの最大数です。

- **pictures** Array of objects
  車両の画像のリストです。

  Array [
  - **title** string
    画像のタイトルです。
  - **url** string <uri>
    画像のURLです。
  - **description** string
    画像の説明文です。

  ]

- **created_at** string <date-time>
  作成日時です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。
- **updated_at** string <date-time>
  最終更新日時です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

]

- **total** integer required
  取得可能な総件数を示します。
- **offset** integer required
  取得する結果の開始位置を示します。デフォルトは0です。
- **limit** integer required
  取得する結果の最大件数を示します。デフォルトは20件です。

> **404** Not Found

> **500** Internal Server Error

➢　リクエストサンプル

```
GET /reservations?passenger_id=user90000001
```

- レスポンスサンプル


```
{
  "reservations": [
    {
      "id": "15912",
      "passenger_id": "user90000001",
      "status": "confirmed",
      "service_id": "odt",
      "pickup": {
        "type": "fixed_stop",
        "stop_id": "stop001",
        "display_name": "",
        "location": {},
        "datetime": "2025-08-19T09:00:00+09:00"
      },
      "dropoff": {
        "type": "fixed_stop",
        "stop_id": "stop002",
        "display_name": "",
        "location": {},
        "datetime": "2025-08-19T10:00:00+09:00"
      },
      "passenger_count": [
        {
          "passenger_type_id": "general",
          "count": 1
        }
      ],
      "accessibility_feature_count": [],
      "fare": {},
      "route_distance": null,
      "vehicle": {
        "id": "vehicle001",
        "name": "車両名",
        "number_plate": "",
        "capacity": {},
        "pictures": [{}]
      },
      "created_at": "2025-08-19T08:00:00+09:00",
      "updated_at": "2025-08-19T08:00:00+09:00"
    }
  ],
  "total": 1,
  "offset": null,
  "limit": null
}
```


【IF025】デマンドバス配車予約検索標準 API

- 本データインターフェースの概要
  - 予約条件に適合したデマンドバスの車両検索を行うためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN006】配車予約管理、【FN010】新規配車予約
- 利用するデータインターフェース
  - 以下のデマンドバス標準インターフェース（2025 年 7 月末受領済の仕様）を利用する。

予約候補を生成

ユーザーが指定した条件（乗車場所・降車場所・希望時間・人数など）をもとに、
対象サービスの中から予約可能な便候補をリアルタイムに生成します。

このAPIを使うことで、アプリ側はユーザーに提示できる具体的な便候補（乗車・降車時刻、運賃）を取得できます。
候補が見つからなかった場合は `candidates` に空配列が返され、`no_candidate_reason` に理由が含まれます。

候補情報は予約確定前の確認用途として設計されており、取得した候補をもとに `/reservations` への予約リクエストが可能です。
`reservation_request` フィールドには、この候補を予約確定する際に `/reservations` にそのまま送信できる予約リクエストのデータが含まれます。
`display` フィールドには、ユーザーに提示するための補足情報（乗降ポイント名、車両名、ナンバープレート、車両画像など）が含まれます。

REQUEST BODY SCHEMA: application/json

service_ids
required

Array of strings
サービスIDのリストです。
複数のサービスで利用できる乗車場所、降車場所を用いる場合、それぞれのサービスに対して予約便候補検索を行うことが可能です。

preferred_time
required

object
希望する乗車または降車の日時を指定します。

type
required

string
Enum: `"pickup"` `"dropoff"`
乗車時刻指定か降車時刻指定かを示します。

- `pickup`
  乗車時刻を指定する場合に選択します。
- `dropoff`
  降車時刻を指定する場合に選択します。

予約候補を生成

ユーザーが指定した条件（乗車場所・降車場所・希望時間・人数など）をもとに、
対象サービスの中から予約可能な便候補をリアルタイムに生成します。

このAPIを使うことで、アプリ側はユーザーに提示できる具体的な便候補（乗車・降車時刻、運賃）を取得できます。
候補が見つからなかった場合は `candidates` に空配列が返され、`no_candidate_reason` に理由が含まれます。

候補情報は予約確定前の確認用途として設計されており、取得した候補をもとに `/reservations` への予約リクエストが可能です。
`reservation_request` フィールドには、この候補を予約確定する際に `/reservations` にそのまま送信できる予約リクエストのデータが含まれます。
`display` フィールドには、ユーザーに提示するための補足情報（乗降ポイント名、車両名、ナンバープレート、車両画像など）が含まれます。

REQUEST BODY SCHEMA: application/json

- **service_ids**
  required
  Array of strings
  サービスIDのリスト です。
  複数のサービスで利用できる乗車場所、降車場所を用いる場合、それぞれのサービスに対して予約便候補検索を行うことが可能です。

- **preferred_time** ⌄
  required
  object
  希望する乗車または降車の日時を指定します。

  - **type**
    required
    string
    Enum: `"pickup"` `"dropoff"`
    乗車時刻指定か降車時刻指定かを示します。

    - `pickup`
      乗車時刻を指定する場合に選択します。
    - `dropoff`
      降車時刻を指定する場合に選択します。

One of | object | object | object

type
required
string
Value: `"custom_location"`
乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

location >
required
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

One of | object | object | object

type
required
string
Value: `"home"`
乗降場所の種類を示します。`home` の場合、自宅を示します。

passenger_id
required
string
ユーザーの自宅位置を指定する場合のユーザーIDです。

dropoff
required

object or object or object
固定の乗降場所・任意の地点・または自宅のいずれかを指定するための情報です。
`stop_id`、`location`、または`passenger_id`のいずれか一つを指定します。

この構造は、予約リクエスト時の入力だけでなく、予約候補や予約情報のレスポンスにも使用されます。
リクエストにおいては、対象サービスがサポートする以下の`reservable_location_types`に応じて、指定できる項目が異なります：

- `fixed_stop`を含むサービスでは、`stop_id`を指定できます。
- `custom_location`を含むサービスでは、`location`を指定できます。
- `home`を含むサービスでは、`passenger_id`を指定することで自宅情報が使用されます。

One of **object** object object

type
required

string
Value: `"fixed_stop"`
乗降場所の種類を示します。`fixed_stop`の場合、固定の乗降場所を示します。

stop_id
required

string
固定の乗降場所のIDです。

One of object **object** object

type
required

string
Value: `"custom_location"`
乗降場所の種類を示します。`custom_location`の場合、任意の地点を示します。

location >
required

object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

One of　object　object　object

- **type** (required)
  string
  Value: `"home"`
  乗降場所の種類を示します。`home` の場合、自宅を示します。
- **passenger_id** (required)
  string
  ユーザーの自宅位置を指定する場合のユーザーIDです。

- **passenger_count** (required)
  Array of objects　non-empty
  乗客の種別ごとの人数を示すリストです。
  乗客種別は、サービスの詳細情報（GET /v1/services/{id}）で取得できる`available_passenger_types`を参照してください。
  合計人数が1人以上になるように指定してください。 0人の種別は指定できません。

Array (non-empty) [

- **passenger_type_id** (required)
  string
  乗客の種別です。
- **count** (required)
  integer <int32>　>= 1
  この種別の乗客数です。

]

accessibility_feature_count
required

Array of objects >= 0 items
乗客全体のうち、補助機能を必要とする人数を種類ごとに指定します。
各人数は `passenger_count` に含まれているものとして扱います（内数）。
補助機能を利用する乗客が誰かは問わず、「何人分必要か」を表すための項目です。
不要な場合は空配列（[]）を指定してください。
種類はサービス詳細（GET /v1/services/{id}）の `supported_accessibility_features` を参照してください。

Array (>= 0 items) [

- **type**
  required
  string
  乗客補助機能の種類です。利用可能な種類はサービスの詳細情報（GET /v1/services/{id}）で取得できる `supported_accessibility_features` を参照してください。
- **count**
  required
  integer <int32> >= 1
  この種類の乗客補助機能の必要数です。

]

vehicle_id

string
車両IDです。
特定の車両を指定して便候補を検索したい場合には指定してください。

Responses

200 OK

RESPONSE SCHEMA: application/json

- **candidates** (required) — Array of objects, >= 0 items
  予約便候補のリストです。
  候補が見つからなかった場合は空の配列が返されます。
  候補がある場合は、各候補の詳細情報が含まれます。

Array (>= 0 items) [

- **reservation_request** (required) — object
  予約便候補の詳細情報です。
  - **candidate_id** (required) — string
    予約便候補の一意な識別子です。
  - **service_id** (required) — string
    予約便候補が属するサービスの一意な識別子です。
  - **pickup** (required) — object or object or object
    乗車場所と予定時刻の情報です。

    One of: object | object | object

One of | object | object | object

type (required)
string
Value: `"fixed_stop"`
乗降場所の種類を示します。`fixed_stop` の場合、固定の乗降場所を示します。

stop_id (required)
string
固定の乗降場所のIDです。`type` が `fixed_stop` の場合のみ設定されます。

location (required)
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

- **type** (required)
  string
  Value: `"Point"`
  GeoJSONにおけるジオメトリタイプ（常に `Point`）
- **coordinates** (required)
  Array of numbers <double> = 2 items [items <double >]
  経度・緯度を含む配列です。
  `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime (required)
string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

type
required
string
Value: `"custom_location"`
乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

location
required
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

- **type**
  required
  string
  Value: `"Point"`
  GeoJSONにおけるジオメトリタイプ（常に `Point`）
- **coordinates**
  required
  Array of numbers <double> = 2 items [items <double >]
  経度・緯度を含む配列です。
  `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required
string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of　object　object　object

type
required
string
Value: `"home"`
乗降場所の種類を示します。`home` の場合、自宅を示します。

passenger_id
required
string
自宅の位置を指定する場合のユーザーIDです。`type` が `home` の場合のみ設定されます。

location
required
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required
string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required
Array of numbers <double>　= 2 items　[items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required
string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。


dropoff
required
object or object or object
降車場所と予定時刻の情報です。
One of
object
object
object
type
required
string
Value: "fixed_stop"
乗降場所の種類を示します。fixed_stop の場合、固定の乗降場所を示します。
stop_id
required
string
固定の乗降場所のIDです。type が fixed_stop の場合のみ設定されます。
location
required
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。
datetime
required
string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

type *required*
string
Value: `"custom_location"`
乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

location *required*
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

- **type** *required*
  string
  Value: `"Point"`
  GeoJSONにおけるジオメトリタイプ（常に `Point`）
- **coordinates** *required*
  Array of numbers <double> = 2 items [items <double >]
  経度・緯度を含む配列です。
  `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime *required*
string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

type
required
string
Value: `"home"`
乗降場所の種類を示します。`home` の場合、自宅を示します。

passenger_id
required
string
自宅の位置を指定する場合のユーザーIDです。`type` が `home` の場合のみ設定されます。

location
required
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

- **type**
  required
  string
  Value: `"Point"`
  GeoJSONにおけるジオメトリタイプ（常に `Point`）
- **coordinates**
  required
  Array of numbers <double> = 2 items [items <double >]
  経度・緯度を含む配列です。
  `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required
string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

- `passenger_count` required
  Array of objects non-empty
  乗客の種別ごとの人数を示すリストです。

  Array (non-empty) [
  - `passenger_type_id` required
    string
    乗客の種別です。
  - `count` required
    integer <int32> >= 1
    この種別の乗客数です。

  ]

- **`accessibility_feature_count`** required
  Array of objects >= 0 items
  利用する乗客補助機能の種類と数です。

  Array (>= 0 items) [
  - `type` required
    string
    乗客補助機能の種類です。利用可能な種類はサービスの詳細情報（GET /v1/services/{id}）で取得できる `supported_accessibility_features` を参照してください。
  - `count` required
    integer <int32> >= 1
    この種類の乗客補助機能の必要数です。

  ]

- `vehicle_id`
  string
  車両IDです。

display
required
object
予約便候補の表示用情報です。

pickup
required
object

display_name
required
string
乗車場所の表示名です。

dropoff
required
object

display_name
required
string
降車場所の表示名です。

fare
required
object
運賃の情報です。

route_distance
integer <int32>
乗車地点から降車地点までのルート探索に基づく推定距離（メートル）です。
直線距離ではなく、道路網上の最短ルートを基に計算された値です。

vehicle
object
この予約で利用する車両の情報です。
vehicle に値があるときは `id` と `name` は必ず値が設定されます。

- id — string
  車両IDです。
- name — string
  車両名です。
- number_plate — string
  車両のナンバープレート番号です。
- capacity — object
  車両の座席数と乗客補助機能の情報です。
  各車両は、最大座席数と利用可能な乗客補助機能の種類とその最大数を持ちます。
  - seats — integer <int32> >= 1
    車両の最大座席数です。
  - accessibility_features — Array of objects
    車両に備わっている乗客補助機能の種類とそれぞれの最大数です。
- **pictures** — Array of objects
  車両の画像のリストです。

Array [
- title — string
  画像のタイトルです。
- url — string <uri>
  画像のURLです。
- description — string
  画像の説明文です。

]

no_candidate_reason string

Enum: `"no_operation_plans"` `"stops_is_suspended"` `"location_out_of_reservable_areas"` `"no_candidate"`

候補が見つからなかった場合の理由（候補がある場合はnull）です。

- `no_operation_plans`
  運行が存在しない
- `stops_is_suspended`
  乗降場所が利用停止中
- `location_out_of_reservable_areas`
  指定された乗降場所が予約可能エリア外
- `no_candidate`
  その他の理由で候補がない

> **400** Bad Request

> **500** Internal Server Error

➢　リクエストサンプル

```
{
  "service_ids": ["odt"],
  "preferred_time": {
    "type": "pickup",
    "datetime": "2024-01-15T09:00:00+09:00"
  },
  "pickup": {
    "type": "fixed_stop",
    "stop_id": "stop001"
  },
  "dropoff": {
    "type": "fixed_stop",
    "stop_id": "stop002"
  },
  "passenger_count": [
    {
      "passenger_type_id": "general",
      "count": 2
    }
  ],
  "accessibility_feature_count": []
}
```

➢　レスポンスサンプル

```
{
  "candidates": [
    {
      "reservation_request": {
        "candidate_id": "12345",
        "service_id": "odt",
        "pickup": {
          "type": "fixed_stop",
          "stop_id": "stop001",
          "location": {},
          "datetime": "2024-01-15T09:00:00+09:00"
        },
        "dropoff": {
          "type": "fixed_stop",
          "stop_id": "stop002",
          "location": {},
          "datetime": "2024-01-15T10:00:00+09:00"
        },
        "passenger_count": [
          {
            "passenger_type_id": "general",
            "count": 2
          }
        ],
        "accessibility_feature_count": [],
        "vehicle_id": "vehicle001"
      },
      "display": {
        "pickup": {
          "display_name": ""
        },
        "dropoff": {
          "display_name": ""
        },
        "fare": {},
        "route_distance": null,
        "vehicle": {}
      },
      "option": {
        "total_duration_change": 0
      }
    }
  ],
  "no_candidate_reason": null
}
```


【IF026】デマンドバス配車予約標準 API

- 本データインターフェースの概要
  - デマンドバスの配車予約を行うためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN006】配車予約管理、【FN010】新規配車予約
- 利用するデータインターフェース
  - 以下のデマンドバス標準インターフェース（2025 年 7 月末受領済の仕様）を利用する。

予約を登録

事前の検索によって取得された予約便候補を用いて、予約を登録します。

このエンドポイントでは、事前に `/reservations/candidates` で取得した `candidate_id` およびそのときの `reservation_request` データを使用して、予約を登録します。
`candidate_id` は必須で、それ以外の項目（pickup, dropoff, fare など）は候補取得時の内容との照合・検証を目的としています。
一致しない場合、リクエストはエラーとして拒否されます。候補取得時のデータをそのまま使用することを前提とした設計になります。

REQUEST BODY SCHEMA: application/json

passenger_id required	string 予約を登録するユーザーの一意な識別子です。
candidate_id required	string 予約便候補の一意な識別子です。
service_id required	string 予約便候補が属するサービスの一意な識別子です。
pickup ⌄ required	object or object or object 乗車場所と予定時刻の情報です。

One of object object object

type
required
string
Value: `"fixed_stop"`
乗降場所の種類を示します。`fixed_stop` の場合、固定の乗降場所を示します。

stop_id
required
string
固定の乗降場所のIDです。`type` が `fixed_stop` の場合のみ設定されます。

location
required
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required
string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required
Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required
string <date-time>
datetime または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

type *required*
string
Value: `"custom_location"`
乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

location *required*
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

- **type** *required*
  string
  Value: `"Point"`
  GeoJSONにおけるジオメトリタイプ（常に `Point`）
- **coordinates** *required*
  Array of numbers <double> = 2 items [items <double >]
  経度・緯度を含む配列です。
  `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime *required*
string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

- **type** (required)
  string
  Value: `"home"`
  乗降場所の種類を示します。`home` の場合、自宅を示します。

- **passenger_id** (required)
  string
  自宅の位置を指定する場合のユーザーIDです。`type` が `home` の場合のみ設定されます。

- **location** (required)
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

  - **type** (required)
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）

  - **coordinates** (required)
    Array of numbers <double> = 2 items [items <double >]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

- **datetime** (required)
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

dropoff
required

object or object or object
降車場所と予定時刻の情報です。

One of object | object | object

type
required

string
Value: `"fixed_stop"`
乗降場所の種類を示します。`fixed_stop` の場合、固定の乗降場所を示します。

stop_id
required

string
固定の乗降場所のIDです。`type` が `fixed_stop` の場合のみ設定されます。

location
required

object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required

string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required

Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required

string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

dropoff
required

object or object or object
降車場所と予定時刻の情報です。

One of object | object | object

type
required

string
Value: `"custom_location"`
乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

location
required

object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

location

type
required

string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required

Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required

string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

dropoff required
object or object or object
降車場所と予定時刻の情報です。

One of | object | object | object

type required
string
Value: `"home"`
乗降場所の種類を示します。`home` の場合、自宅を示します。

passenger_id required
string
自宅の位置を指定する場合のユーザーIDです。`type` が `home` の場合のみ設定されます。

location required
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

- **type** required
  string
  Value: `"Point"`
  GeoJSONにおけるジオメトリタイプ（常に `Point`）
- **coordinates** required
  Array of numbers <double> = 2 items [items <double >]
  経度・緯度を含む配列です。
  `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime required
string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

- `passenger_count` required
  Array of objects non-empty
  乗客の種別ごとの人数を示すリストです。

  `Array (non-empty) [`

  - `passenger_type_id` required
    string
    乗客の種別です。
  - `count` required
    integer <int32> >= 1
    この種別の乗客数です。

  `]`

- **`accessibility_feature_count`** required
  Array of objects >= 0 items
  利用する乗客補助機能の種類と数です。

  `Array (>= 0 items) [`

  - `type` required
    string
    乗客補助機能の種類です。利用可能な種類はサービスの詳細情報（GET /v1/services/{id}）で取得できる `supported_accessibility_features` を参照してください。
  - `count` required
    integer <int32> >= 1
    この種類の乗客補助機能の必要数です。

  `]`

- `vehicle_id`
  string
  車両IDです。

Responses

201 Created

RESPONSE SCHEMA: application/json

id
required
string
登録済み予約の一意な識別子です。

passenger_id
required
string
予約を登録したユーザーの一意な識別子です。

status
required
string
Enum: `"tentative"` `"confirmed"` `"no_show"` `"in_transit"` `"completed"` `"cancelled"`
予約のステータスです。

status

- `tentative`
  仮予約。乗車便が未確定で、締切前の状態です。確定処理が行われるまではこの状態となります。
- `confirmed`
  確定予約。 確定予約。乗車便が成立した状態です。
- `in_transit`
  乗車中。車両が運行中で、乗客が乗車している状態です。
- `completed`
  降車済み。乗車が完了し、正常に終了した予約です。
- `no_show`
  未乗車。予約は確定していたものの、乗客が現れず乗車しなかった状態です。
- `cancelled`
  キャンセル済み。利用者またはシステムにより予約が取り消された状態です。（仮予約・確定予約いずれからも遷移します）

service_id
required
string
予約便候補が属するサービスの一意な識別子です。

pickup
required
object or object or object
乗車場所と予定時刻の情報です。

One of object object object

type
required
string
Value: "fixed_stop"
乗降場所の種類を示します。fixed_stop の場合、固定の乗降場所を示します。

stop_id
required
string
固定の乗降場所のIDです。type が fixed_stop の場合のみ設定されます。

display_name
required
string
乗降場所の表示名です。type が fixed_stop の場合、乗降場所の名称が設定されます。

location
required
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required
string
Value: "Point"
GeoJSONにおけるジオメトリタイプ（常に Point ）

coordinates
required
Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
[longitude, latitude] の順で格納されます（GeoJSON仕様準拠）。

datetime
required
string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

type *required*
string
Value: `"custom_location"`
乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

display_name *required*
string
乗降場所の表示名です。`type` が `custom_location` の場合「任意地点」が設定されます。

location *required*
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

- **type** *required*
  string
  Value: `"Point"`
  GeoJSONにおけるジオメトリタイプ（常に `Point`）
- **coordinates** *required*
  Array of numbers <double> = 2 items [items <double>]
  経度・緯度を含む配列です。
  `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime *required*
string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

- **type** (required)
  string
  Value: `"home"`
  乗降場所の種類を示します。`home` の場合、自宅を示します。
- **passenger_id** (required)
  string
  自宅の位置を指定する場合のユーザーIDです。`type` が `home` の場合のみ設定されます。
- **display_name** (required)
  string
  乗降場所の表示名です。`type` が `home` の場合「自宅」が設定されます。
- **location** (required)
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。
  - **type** (required)
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）
  - **coordinates** (required)
    Array of numbers <double> = 2 items [items <double>]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。
- **datetime** (required)
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

dropoff
required

object or object or object
降車場所と予定時刻の情報です。

One of object | object | object

type
required

string
Value: `"fixed_stop"`
乗降場所の種類を示します。`fixed_stop` の場合、固定の乗降場所を示します。

stop_id
required

string
固定の乗降場所のIDです。`type` が `fixed_stop` の場合のみ設定されます。

display_name
required

string
乗降場所の表示名です。`type` が `fixed_stop` の場合、乗降場所の名称が設定されます。

location
required

object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required

string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required

Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required

string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

dropoff
required

object or object or object
降車場所と予定時刻の情報です。

One of | object | object | object

type
required

string
Value: `"custom_location"`
乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

display_name
required

string
乗降場所の表示名です。`type` が `custom_location` の場合「任意地点」が設定されます。

location
required

object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required

string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required

Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required

string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

- **type** (required)
  string
  Value: `"home"`
  乗降場所の種類を示します。`home` の場合、自宅を示します。

- **passenger_id** (required)
  string
  自宅の位置を指定する場合のユーザーIDです。`type` が `home` の場合のみ設定されます。

- **display_name** (required)
  string
  乗降場所の表示名です。`type` が `home` の場合「自宅」が設定されます。

- **location** (required)
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

  - **type** (required)
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）

  - **coordinates** (required)
    Array of numbers <double> = 2 items [items <double >]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

- **datetime** (required)
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

passenger_count
required

Array of objects non-empty
乗客の種別ごとの人数を示すリストです。

Array (non-empty) [

passenger_type_id
required

string
乗客の種別です。

count
required

integer <int32> >= 1
この種別の乗客数です。

]

accessibility_feature_count
required

Array of objects >= 0 items
利用する乗客補助機能の種類と数です。

Array (>= 0 items) [

type
required

string
乗客補助機能の種類です。利用可能な種類はサービスの詳細情報（GET /v1/services/{id}）で取得できる `supported_accessibility_features` を参照してください。

count
required

integer <int32> >= 1
この種類の乗客補助機能の必要数です。

]

fare
required

object
運賃の情報です。

per_passenger_type
required

Array of objects >= 0 items
乗客の種別ごとの運賃小計を示すリストです。

total

integer <int32> >= 0
運賃の合計（円）です。

route_distance
integer <int32>
乗車地点から降車地点までのルート探索に基づく推定距離（メートル）です。
直線距離ではなく、道路網上の最短ルートを基に計算された値です。

vehicle
object
この予約で利用する車両の情報です。
vehicle に値があるときは `id` と `name` は必ず値が設定されます。

id
string
車両IDです。

name
string
車両名です。

number_plate
string
車両のナンバープレート番号です。

capacity
object
車両の座席数と乗客補助機能の情報です。
各車両は、最大座席数と利用可能な乗客補助機能の種類とその最大数を持ちます。
capacity

seats
integer <int32> >= 1
車両の最大座席数です。

accessibility_features
Array of objects
車両に備わっている乗客補助機能の種類とそれぞれの最大数です。

pictures Array of objects
車両の画像のリストです。

Array [

- **title** string
  画像のタイトルです。
- **url** string <uri>
  画像のURLです。
- **description** string
  画像の説明文です。

]

- **created_at** string <date-time>
  作成日時です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。
- **updated_at** string <date-time>
  最終更新日時です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

> **400** Bad Request

> **409** Conflict

> **500** Internal Server Error

- リクエストサンプル


```
{
  "candidate_id": "12345",
  "passenger_id": "user001",
  "service_id": "odt",
  "pickup": {
    "type": "fixed_stop",
    "stop_id": "stop001",
    "display_name": "出発地",
    "location": {},
    "datetime": "2024-01-15T09:00:00+09:00"
  },
  "dropoff": {
    "type": "fixed_stop",
    "stop_id": "stop002",
    "display_name": "到着地",
    "location": {},
    "datetime": "2024-01-15T10:00:00+09:00"
  },
  "passenger_count": [
    {
      "passenger_type_id": "general",
      "count": 2
    }
  ],
  "accessibility_feature_count": []
}
```


- レスポンスサンプル


```
{
  "id": "15912",
  "passenger_id": "user001",
  "status": "confirmed",
  "service_id": "odt",
  "pickup": {
    "type": "fixed_stop",
    "stop_id": "stop001",
    "display_name": "",
    "location": {},
    "datetime": "2024-01-15T09:00:00+09:00"
  },
  "dropoff": {
    "type": "fixed_stop",
    "stop_id": "stop002",
    "display_name": "",
    "location": {},
    "datetime": "2024-01-15T10:00:00+09:00"
  },
  "passenger_count": [
    {
      "passenger_type_id": "general",
      "count": 2
    }
  ],
  "accessibility_feature_count": [],
  "fare": {},
  "route_distance": null,
  "vehicle": {
    "id": "vehicle001",
    "name": "車両名",
    "number_plate": "",
    "capacity": {},
    "pictures": []
  },
  "created_at": "2024-01-15T09:00:00+09:00",
  "updated_at": "2024-01-15T09:00:00+09:00"
}
```

【IF027】ゲートウェイー配車予約変更連携 IF ＜新規開発＞

- 本データインターフェースの概要
  - 配車予約変更標準 API にて連携された内容を配車予約変更機能に連携し処理を引き継ぐ。
- 本インターフェースを利用する機能
  - 【FN009】ゲートウェイ、【FN011】配車予約変更
- データ詳細
  - 入力

名称	説明	値	必須
user_id	利用者番号	String	○
reservation_ids	変更対象の予約 ID（Array 型）	Number	○
arrival_demand_time	変更になった病院到着希望日時	Time ISO8601 形式	○
departure_demand_time	変更になった病院出発希望日時	Time ISO8601 形式	○
way_points_id	立ち寄り先 ID	Number	―
stay_duration_minutes	立ち寄り先の滞在時間	Number	―

【IF028】配車予約変更ー配車キャンセル連携 IF ＜新規開発＞

- 本データインターフェースの概要
  - 配車予約変更標準 API 経由で連携されたキャンセルリクエスト情報を、配車予約キャンセル機能に連携する。
- 本インターフェースを利用する機能
  - 【FN011】配車予約変更、【FN012】配車予約キャンセル
- データ詳細
  - 入力

名称	説明	値	必須
user_id	利用者番号	String	○
reservation_ids	変更対象の予約 ID（Array 型）、最大 3 個	Number	○

【IF029】配車予約変更ー新規配車予約連携 IF ＜新規開発＞

- 本データインターフェースの概要
  - 配車予約変更標準 API 経由で連携された配車予約情報（変更情報）を、新規配車予約機能に連携する。
- 本インターフェースを利用する機能
  - 【FN011】配車予約変更、【FN010】新規配車予約
- データ詳細
  - 入力

名称	説明	値	必須
user_id	利用者番号	String	○
arrival_demand_time	病院到着希望日時	Time ISO8601 形式	○-
departure_demand_time	病院出発希望日時	Time ISO8601 形式	○

【IF030】ゲートウェイー配車予約キャンセル連携 IF ＜新規開発＞

- 本データインターフェースの概要
  - 配車予約キャンセル標準 API 経由で連携されたキャンセルリクエスト情報を配車予約キャンセル機能に連携する。
- 本インターフェースを利用する機能
  - 【FN009】ゲートウェイ、【FN012】配車予約キャンセル
- データ詳細
  - 入力

名称	説明	値	必須
user_id	利用者番号	String	○
reservation_ids	変更対象の予約 ID（Array 型）、最大 3 個	Number	○

【IF031】デマンドバス配車予約キャンセル標準 API

- 本データインターフェースの概要
  - デマンドバスの配車予約キャンセルを行うためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN006】配車予約管理、【FN012】配車予約キャンセル
- データ詳細
  - 以下に詳細を記載する

予約を更新

登録済みの予約情報を予約ID指定で更新します。
予約のキャンセルはこのAPIを用いて、ステータスを `cancelled` に更新することで行います。

更新可能な項目は、以下の項目のみです。

- `passenger_count`（乗客の種別ごとの人数）
- `accessibility_feature_count`（乗客補助機能の種類と数）
- `status`（予約のステータス）
  予約のステータスは以下の変更のみが可能です。
  - `tentative` から `cancelled`
  - `confirmed` から `cancelled`

PATH PARAMETERS

id required	string Example: `1` 予約IDを指定します。

REQUEST BODY SCHEMA: application/json

passenger_id required	string 予約を登録したユーザーの一意な識別子です。

status
required

string
Enum: `"tentative"` `"confirmed"` `"no_show"` `"in_transit"` `"completed"` `"cancelled"`
予約のステータスです。

- `tentative`
  仮予約。乗車便が未確定で、締切前の状態です。確定処理が行われるまではこの状態となります。
- `confirmed`
  確定予約。 確定予約。乗車便が成立した状態です。
- `in_transit`
  乗車中。車両が運行中で、乗客が乗車している状態です。
- `completed`
  降車済み。乗車が完了し、正常に終了した予約です。
- `no_show`
  未乗車。予約は確定していたものの、乗客が現れず乗車しなかった状態です。
- `cancelled`
  キャンセル済み。利用者またはシステムにより予約が取り消された状態です。（仮予約・確定予約いずれからも遷移します）

---

service_id
required

string
予約便候補が属するサービスの一意な識別子です。

---

pickup
required

object or object or object
乗車場所と予定時刻の情報です。

One of　object　object　object

- **type** (required)
  string
  Value: `"fixed_stop"`
  乗降場所の種類を示します。`fixed_stop` の場合、固定の乗降場所を示します。

- **stop_id** (required)
  string
  固定の乗降場所のIDです。`type` が `fixed_stop` の場合のみ設定されます。

- **display_name** (required)
  string
  乗降場所の表示名です。`type` が `fixed_stop` の場合、乗降場所の名称が設定されます。

- **location** (required)
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

  - **type** (required)
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）

  - **coordinates** (required)
    Array of numbers <double>　= 2 items　[items <double >]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

- **datetime** (required)
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | **object** | object

- **type** (required)
  string
  Value: `"custom_location"`
  乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

- **display_name** (required)
  string
  乗降場所の表示名です。`type` が `custom_location` の場合「任意地点」が設定されます。

- **location** (required)
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

  - **type** (required)
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）

  - **coordinates** (required)
    Array of numbers <double> = 2 items [items <double >]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

- **datetime** (required)
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

- **type** (required)
  string
  Value: `"home"`
  乗降場所の種類を示します。`home` の場合、自宅を示します。

- **passenger_id** (required)
  string
  自宅の位置を指定する場合のユーザーIDです。`type` が `home` の場合のみ設定されます。

- **display_name** (required)
  string
  乗降場所の表示名です。`type` が `home` の場合「自宅」が設定されます。

- **location** (required)
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

  - **type** (required)
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）

  - **coordinates** (required)
    Array of numbers <double> = 2 items [items <double>]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

- **datetime** (required)
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

dropoff
required

object or object or object
降車場所と予定時刻の情報です。

One of object | object | object

type
required

string
Value: `"fixed_stop"`
乗降場所の種類を示します。`fixed_stop` の場合、固定の乗降場所を示します。

stop_id
required

string
固定の乗降場所のIDです。`type` が `fixed_stop` の場合のみ設定されます。

display_name
required

string
乗降場所の表示名です。`type` が `fixed_stop` の場合、乗降場所の名称が設定されます。

location
required

object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required

string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required

Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required

string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

- **type** (required)
  string
  Value: `"custom_location"`
  乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

- **display_name** (required)
  string
  乗降場所の表示名です。`type` が `custom_location` の場合「任意地点」が設定されます。

- **location** (required)
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

  - **type** (required)
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）

  - **coordinates** (required)
    Array of numbers <double> = 2 items [items <double >]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

- **datetime** (required)
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

- **type** (required)
  string
  Value: `"home"`
  乗降場所の種類を示します。`home` の場合、自宅を示します。

- **passenger_id** (required)
  string
  自宅の位置を指定する場合のユーザーIDです。`type` が `home` の場合のみ設定されます。

- **display_name** (required)
  string
  乗降場所の表示名です。`type` が `home` の場合「自宅」が設定されます。

- **location** (required)
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

  - **type** (required)
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）

  - **coordinates** (required)
    Array of numbers <double> = 2 items [items <double>]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

- **datetime** (required)
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

- `passenger_count` required
  Array of objects non-empty
  乗客の種別ごとの人数を示すリストです。

  Array (non-empty) [
  - `passenger_type_id` required
    string
    乗客の種別です。
  - `count` required
    integer <int32> >= 1
    この種別の乗客数です。

  ]

- `accessibility_feature_count` required
  Array of objects >= 0 items
  利用する乗客補助機能の種類と数です。

  Array (>= 0 items) [
  - `type` required
    string
    乗客補助機能の種類です。利用可能な種類はサービスの詳細情報（GET /v1/services/{id}）で取得できる `supported_accessibility_features` を参照してください。
  - `count` required
    integer <int32> >= 1
    この種類の乗客補助機能の必要数です。

  ]

- **`vehicle`**
  object
  この予約で利用する車両の情報です。
  vehicle に値があるときは `id` と `name` は必ず値が設定されます。

id	string	車両IDです。
name	string	車両名です。
number_plate	string	車両のナンバープレート番号です。
capacity >	object	車両の座席数と乗客補助機能の情報です。 各車両は、最大座席数と利用可能な乗客補助機能の種類とその最大数を持ちます。
pictures >	Array of objects	車両の画像のリストです。

Responses

200 OK

RESPONSE SCHEMA: application/json

id required	string	登録済み予約の一意な識別子です。
passenger_id required	string	予約を登録したユーザーの一意な識別子です。

status
required

string
Enum: `"tentative"` `"confirmed"` `"no_show"` `"in_transit"` `"completed"` `"cancelled"`
予約のステータスです。

- `tentative`
  仮予約。乗車便が未確定で、締切前の状態です。確定処理が行われるまではこの状態となります。
- `confirmed`
  確定予約。 確定予約。乗車便が成立した状態です。
- `in_transit`
  乗車中。車両が運行中で、乗客が乗車している状態です。
- `completed`
  降車済み。乗車が完了し、正常に終了した予約です。
- `no_show`
  未乗車。予約は確定していたものの、乗客が現れず乗車しなかった状態です。
- `cancelled`
  キャンセル済み。利用者またはシステムにより予約が取り消された状態です。（仮予約・確定予約いずれからも遷移します）

---

service_id
required

string
予約便候補が属するサービスの一意な識別子です。

---

pickup
required

object or object or object
乗車場所と予定時刻の情報です。

One of **object** | object | object

- **type** *required*
  string
  Value: `"fixed_stop"`
  乗降場所の種類を示します。`fixed_stop` の場合、固定の乗降場所を示します。

- **stop_id** *required*
  string
  固定の乗降場所のIDです。`type` が `fixed_stop` の場合のみ設定されます。

- **display_name** *required*
  string
  乗降場所の表示名です。`type` が `fixed_stop` の場合、乗降場所の名称が設定されます。

- **location** *required*
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

  - **type** *required*
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）

  - **coordinates** *required*
    Array of numbers <double> = 2 items [items <double >]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

- **datetime** *required*
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

- **type** (required)
  string
  Value: `"custom_location"`
  乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

- **display_name** (required)
  string
  乗降場所の表示名です。`type` が `custom_location` の場合「任意地点」が設定されます。

- **location** (required)
  object
  乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

  - **type** (required)
    string
    Value: `"Point"`
    GeoJSONにおけるジオメトリタイプ（常に `Point`）

  - **coordinates** (required)
    Array of numbers <double> = 2 items [items <double >]
    経度・緯度を含む配列です。
    `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

- **datetime** (required)
  string <date-time>
  乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

One of | object | object | object

type *required*
string
Value: `"home"`
乗降場所の種類を示します。`home` の場合、自宅を示します。

passenger_id *required*
string
自宅の位置を指定する場合のユーザーIDです。`type` が `home` の場合のみ設定されます。

display_name *required*
string
乗降場所の表示名です。`type` が `home` の場合「自宅」が設定されます。

location *required*
object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

- **type** *required*
  string
  Value: `"Point"`
  GeoJSONにおけるジオメトリタイプ（常に `Point`）
- **coordinates** *required*
  Array of numbers <double> = 2 items [items <double >]
  経度・緯度を含む配列です。
  `[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime *required*
string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

dropoff
required

object or object or object
降車場所と予定時刻の情報です。

One of object | object | object

type
required

string
Value: `"fixed_stop"`
乗降場所の種類を示します。`fixed_stop` の場合、固定の乗降場所を示します。

stop_id
required

string
固定の乗降場所のIDです。`type` が `fixed_stop` の場合のみ設定されます。

display_name
required

string
乗降場所の表示名です。`type` が `fixed_stop` の場合、乗降場所の名称が設定されます。

location
required

object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required

string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required

Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required

string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

dropoff
required

object or object or object
降車場所と予定時刻の情報です。

One of | object | object | object

type
required

string
Value: `"custom_location"`
乗降場所の種類を示します。`custom_location` の場合、任意の地点を示します。

display_name
required

string
乗降場所の表示名です。`type` が `custom_location` の場合「任意地点」が設定されます。

location
required

object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required

string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required

Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required

string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

dropoff
required

object or object or object
降車場所と予定時刻の情報です。

One of | object | object | object

type
required

string
Value: `"home"`
乗降場所の種類を示します。`home` の場合、自宅を示します。

passenger_id
required

string
自宅の位置を指定する場合のユーザーIDです。`type` が `home` の場合のみ設定されます。

display_name
required

string
乗降場所の表示名です。`type` が `home` の場合「自宅」が設定されます。

location
required

object
乗降場所を示す、経度・緯度（WGS-84）による地理座標です。

type
required

string
Value: `"Point"`
GeoJSONにおけるジオメトリタイプ（常に `Point`）

coordinates
required

Array of numbers <double> = 2 items [items <double >]
経度・緯度を含む配列です。
`[longitude, latitude]` の順で格納されます（GeoJSON仕様準拠）。

datetime
required

string <date-time>
乗車または降車予定時刻です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

passenger_count
required

Array of objects non-empty
乗客の種別ごとの人数を示すリストです。

Array (non-empty) [

- passenger_type_id
  required
  string
  乗客の種別です。
- count
  required
  integer <int32> >= 1
  この種別の乗客数です。

]

accessibility_feature_count
required

Array of objects >= 0 items
利用する乗客補助機能の種類と数です。

Array (>= 0 items) [

- type
  required
  string
  乗客補助機能の種類です。利用可能な種類はサービスの詳細情報（GET /v1/services/{id}）で取得できる `supported_accessibility_features` を参照してください。
- count
  required
  integer <int32> >= 1
  この種類の乗客補助機能の必要数です。

]

fare
required

object
運賃の情報です。

fare

per_passenger_type
required
Array of objects >= 0 items
乗客の種別ごとの運賃小計を示すリストです。

Array (>= 0 items) [

passenger_type_id
required
string
乗客の種別です。

subtotal
required
integer <int32> >= 0
この種別の運賃小計（円）です。

]

total
integer <int32> >= 0
運賃の合計（円）です。

route_distance
integer <int32>
乗車地点から降車地点までのルート探索に基づく推定距離（メートル）です。
直線距離ではなく、道路網上の最短ルートを基に計算された値です。

vehicle
object
この予約で利用する車両の情報です。
vehicle に値があるときは `id` と `name` は必ず値が設定されます。

id
string
車両IDです。

name
string
車両名です。

capacity object
車両の座席数と乗客補助機能の情報です。
各車両は、最大座席数と利用可能な乗客補助機能の種類とその最大数を持ちます。

seats integer <int32> >= 1
車両の最大座席数です。

accessibility_features > Array of objects
車両に備わっている乗客補助機能の種類とそれぞれの最大数です。

pictures Array of objects
車両の画像のリストです。

Array [

title string
画像のタイトルです。

url string <uri>
画像のURLです。

description string
画像の説明文です。

]

created_at string <date-time>
作成日時です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

updated_at string <date-time>
最終更新日時です。形式はRFC3339のdate-time形式文字列（例：2025-06-19T07:00:00+09:00）です。

> **400** Bad Request

> **404** Not Found

> **500** Internal Server Error

- リクエストサンプル


```
{
  "id": "15912",
  "passenger_id": "user001",
  "status": "cancelled",
  "service_id": "odt",
  "pickup": {
    "type": "fixed_stop",
    "stop_id": "stop001",
    "display_name": "出発地",
    "location": {},
    "datetime": "2024-01-15T09:00:00+09:00"
  },
  "dropoff": {
    "type": "fixed_stop",
    "stop_id": "stop002",
    "display_name": "到着地",
    "location": {},
    "datetime": "2024-01-15T10:00:00+09:00"
  },
  "passenger_count": [
    {
      "passenger_type_id": "general",
      "count": 2
    }
  ],
  "accessibility_feature_count": []
}
```


- レスポンスサンプル


```
{
  "id": "15912",
  "passenger_id": "user001",
  "status": "cancelled",
  "service_id": "odt",
  "pickup": {
    "type": "fixed_stop",
    "stop_id": "stop001",
    "display_name": "出発地",
    "location": {},
    "datetime": "2024-01-15T09:00:00+09:00"
  },
  "dropoff": {
    "type": "fixed_stop",
    "stop_id": "stop002",
    "display_name": "到着地",
    "location": {},
    "datetime": "2024-01-15T10:00:00+09:00"
  },
  "passenger_count": [
    {
      "passenger_type_id": "general",
      "count": 2
    }
  ],
  "accessibility_feature_count": [],
  "fare": {},
  "route_distance": null,
  "vehicle": {
    "id": null,
    "name": null,
    "number_plate": "",
    "capacity": {},
    "pictures": []
  },
  "created_at": "2024-01-15T09:00:00+09:00",
  "updated_at": "2024-01-15T09:00:00+09:00"
}
```

【IF032】立ち寄り先取得 IF　＜新規開発＞

- 本データインターフェースの概要
  - 利用者の立ち寄り先情報を取り込むためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN004】配車予約情報処理
- データ詳細

立ち寄り先取込コード（CSV 形式）

立ち寄り先コード	立ち寄り先名称
001	○○薬局
002	△△ドラッグストア
003	××マーケット

【IF033】配車希望有無入力 IF＜新規開発＞

- 本データインターフェースの概要
- 診療予約に基づく配車予約の希望有無をデータベースに更新するためのインターフェースである。
  - 【FN005】ユーザーインターフェース
- データ詳細

論理名	物理名	型(サイズ)
利用者番号	user_id	String
配車希望有無	reservation_onoff_flag	Boolean

【IF034】配車希望有無取得 IF＜新規開発＞

- 本データインターフェースの概要
  - 診療予約に基づく配車予約の希望有無を PHR アプリに表示するためのインターフェースである。
- 本インターフェースを利用する機能
  - 【FN005】ユーザーインターフェース
- データ詳細

論理名	物理名	型(サイズ)
利用者番号	user_id	String
配車希望有無	reservation_onoff_flag	Boolean

2-5. ユーザーインターフェース（UI）

2-5-1. 画面遷移図

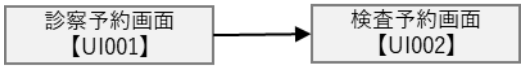


図 2-31 【HW005】PC（PHR システム/電子カルテシステム）用画面遷移図

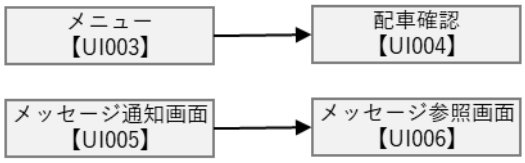


図 2-32 【HW006】スマートフォン（PHR アプリ）用画面遷移図

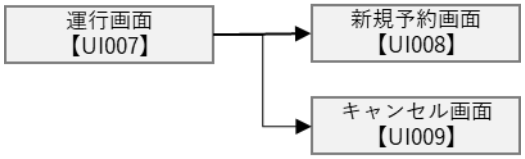


図 2-33 【HW007】PC（デマンドバス配車システム）用画面遷移図

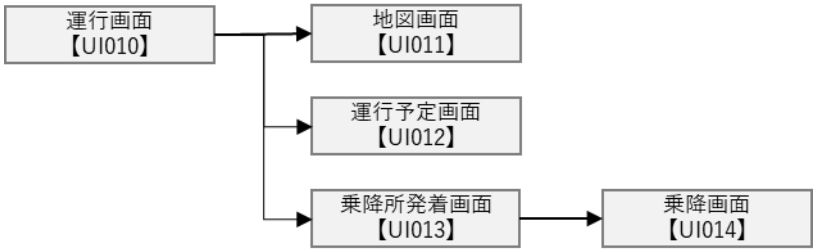


図 2-34【HW008】タブレット（デマンドバス配車システム（ドライバーアプリ））用画面遷移図

2-5-2. ユーザーインターフェース一覧

表 2-12 【HW005】PC（電子カルテシステム）用画面一覧

※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	電子カルテ／診察予約画面	● 医師が診察している患者の次回の外来予約を作成する。	FN001
UI002	電子カルテ／検査予約画面	● 医師が診察している患者に対して検査予約を行う。	FN001

表 2-13 【HW006】スマートフォン（PHR アプリ）用画面一覧

※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）
UI003	メニュー（SP）	● 診察予約や病院情報等の閲覧メニュー（既存サービス）に加えて配車情報に関するメニュー（新規開発）を表示する。	FN005
UI004	配車確認（SP）	● 利用者の配車希望有無を表示、変更を行う。 ● 次回の診察予約、離院予測時刻、行き・帰りの送迎予約内容の確認を行う。 ● 立ち寄り先、立ち寄り時間の候補を表示し、立ち寄り先、立ち寄り時間を選択する。	FN005
UI005	メッセージ通知画面（SP）	● 当日の帰りの送迎又は次回の行き・帰りの送迎予約内容の通知／予約不可通知が Android または iOS の通知機能で乗客のスマホに表示する。	FN005
UI006	メッセージ参照画面（SP）	● 当日の帰りの送迎または次回の行き・帰りの送迎予約内容を表示する。	FN005

表 2-14 【HW007】PC（デマンドバス配車システム）用画面一覧

※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）
UI007	運行画面	● 運行状況を表示する。	FN011

UI008	新規予約画面	● 予約の新規登録を行う。	FN011
UI009	キャンセル画面	● 予約のキャンセルを行う。	FN011

表 2-15 【HW008】タブレット（デマンドバス配車システム（ドライバーアプリ））用画面一覧

※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）
UI010	運行画面	● 次の目的地、到着時刻など現在の運行状況を表示する。	FN010
UI011	地図画面	● 現在地から目的地までの道のりを地図上に表示し、ナビゲーションする。	FN010
UI012	運行予定画面	● 運行予定の乗降場と乗客を表示する。	FN010
UI013	乗降場発着画面	● 乗降場の発着状況の確認を行う。	FN010
UI014	乗降画面	● 利用者の乗降状況の確認を行う。	FN010

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。

1)　【HW005】PC（電子カルテシステム）用画面

【UI001】電子カルテ／診察予約画面

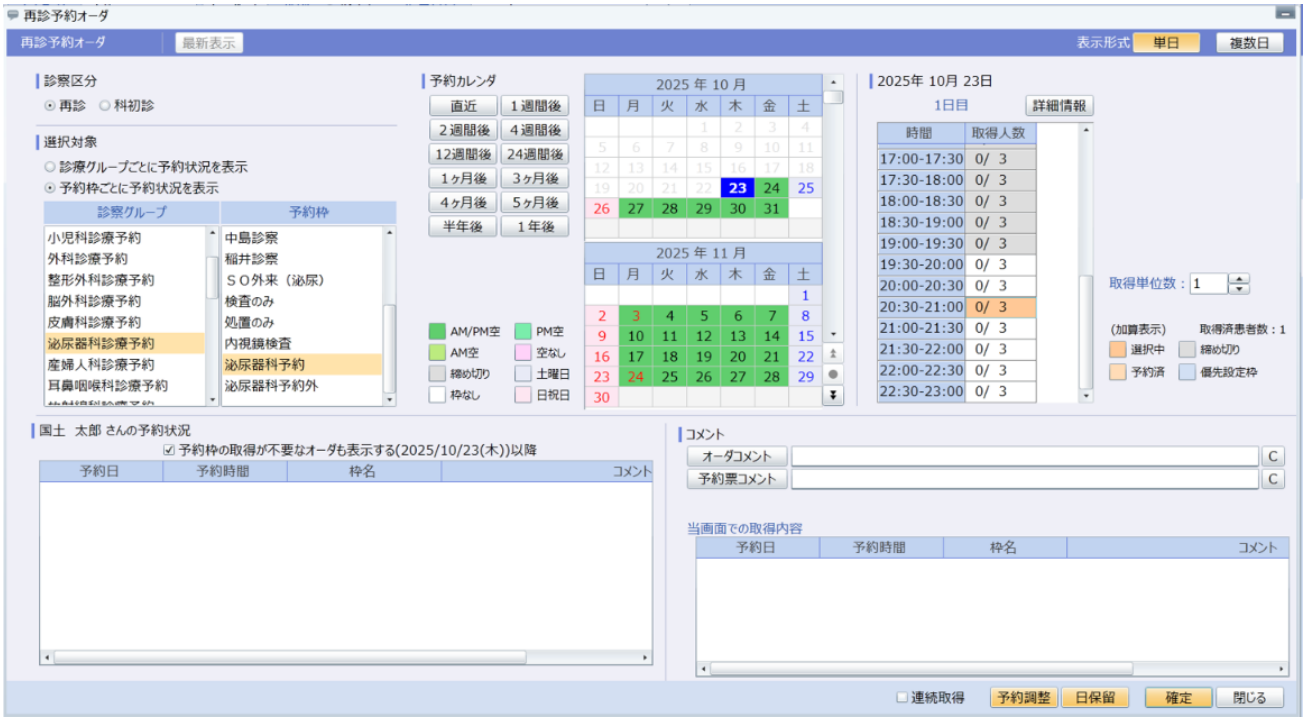


図 2-35　電子カルテ診察予約画面

- 本画面の概要
  - 外来患者の次回診察予約をするための予約枠や日時を表示する。
  - 予約済みの予約日時を確認する。
- 本画面から利用する機能
  - 【FN001】診察予約

【UI002】電子カルテ／検査予約画面

図 2-36　電子カルテ検査予約画面

- 本画面の概要
  - 外来患者の次回診察時の検査予約をするための検査内容を表示する。
  - 患者の病名情報や身長・体重などの生理的情報を表示する。
- 本画面から利用する機能
  - 【FN001】診察予約

2)　【HW001】スマートデバイス用画面

【UI003】メニュー画面＜新規開発＞

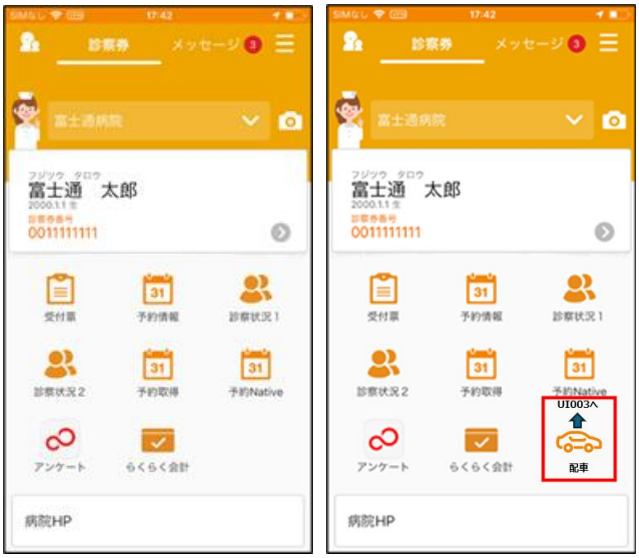


図 2-37　メニュー画面

- 本画面の概要
  - ➢　デマンドバスの配車予約情報などを参照するためのボタンが配備されたメニューを表示する。
- 本画面から利用する機能
  - ➢　【FN005】ユーザーインターフェース

【UI004】配車確認＜新規開発＞

図 2-38　配車確認・変更イメージ図

- 本画面の概要
  - ➢　利用者の配車希望有無を選択できる。
  - ➢　利用者のデマンドバス配車予約情報を表示する。
  - ➢　離院後の立ち寄り先を選択する。
  - ➢　配車予約を変更できる。

- 配車予約をキャンセルできる。

- 本画面から利用する機能
  - 【FN005】ユーザーインターフェース

【UI005】メッセージ通知画面＜新規開発＞

図 2-39　メッセージ通知

- 本画面の概要
  - 利用者が予約した診察日に対する予約情報が更新された場合にスマートデバイスの Push 通知にて表示する。
  - 利用者が予約した診察日の当日に予約情報をスマートデバイスの Push 通知にて表示する。

- 本画面から利用する機能
  - 【FN005】ユーザーインターフェース

【UI006】メッセージ参照画面＜新規開発＞

図 2-40　ッセージ参照画面

- 本画面の概要
  - 利用者が予約した診察日に対する予約情報が更新された場合に、その更新内容を表示する。
  - 利用者が予約した診察日の当日に予約情報を表示する。
- 本画面から利用する機能
  - 【FN005】ユーザーインターフェース

3)　【HW007】PC（デマンドバス配車システム）

【UI007】運行画面

図 2-41　運行画面

- 本画面の概要
  - デマンドバス配車システムに登録された予約情報を日ごとに表示する。
- 本画面から利用する機能
  - 【FN009】配車予約管理

【UI008】新規予約画面

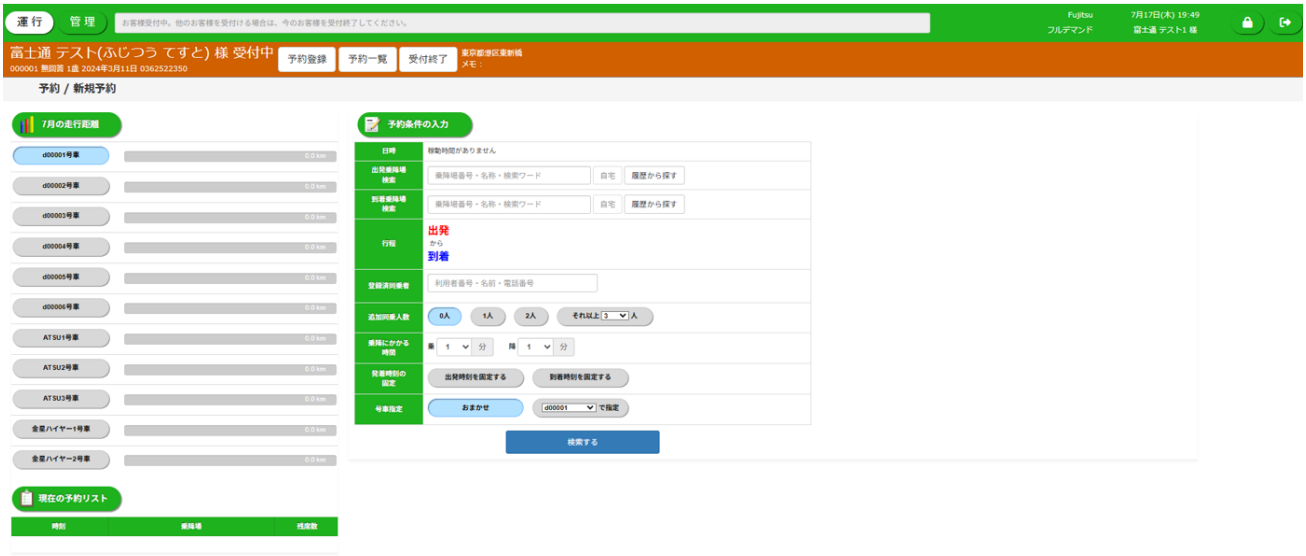


図 2-42　新規予約画面

- 本画面の概要
  - ➢ デマンドバス配車システムに予約を新規登録する。
- 本画面から利用する機能
  - ➢ 【FN011】配車状況確認・配車変更

【UI009】キャンセル画面

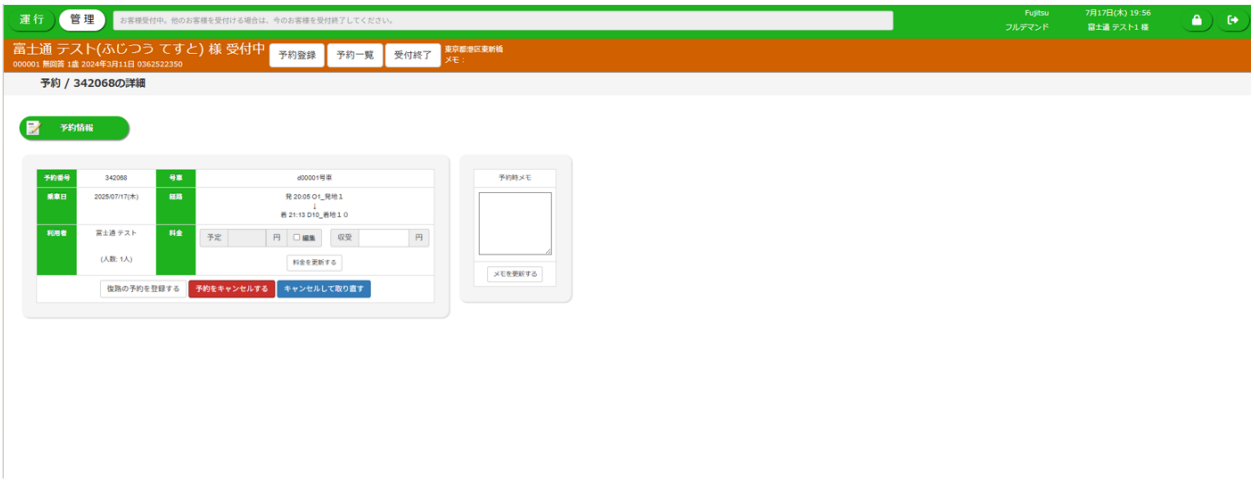


図 2-43　キャンセル画面

- 本画面の概要
  - ➢ デマンドバス配車システムに登録されている予約をキャンセルする。

- 本画面から利用する機能
  - 【FN011】配車状況確認・配車変更

4)　【HW008】タブレット（デマンドバス配車システム（ドライバーアプリ））

【UI010】運行画面

図 2-44　運行画面

- 本画面の概要
  - デマンドバス配車システムに登録された予約情報を基に運行予定を表示する。
  - 【UI011 地図画面、【UI012 運行予定画面、【UI013 乗降発着画面に遷移可能である。
- 本画面から利用する機能
  - 【FN010】運行指示

【UI011】地図画面

図 2-45　図画面

- 本画面の概要
  - 地図を表示し行先に向けたナビゲーションを行う。
- 本画面から利用する機能

- 【FN010】運行指示

【UI012】運行予定画面

図 2-46　行予定画面

- 本画面の概要
  - 直近の運行予定を表示する。
- 本画面から利用する機能
  - 【FN010】運行指示

【UI013】乗降場発着画面

図 2-47　降場発着画面

- 本画面の概要
  - 乗降場の発着を入力する。
- 本画面から利用する機能
  - 【FN010】運行指示

【UI014】乗降画面

図 2-48　降画面

- 本画面の概要
  - 乗降状況の管理を行う。
- 本画面から利用する機能
  - 【FN010】運行指示

3. ヘルスケア MaaS システム：非機能要件（NF）

3-1. 非機能要件一覧

非機能要件一覧について以下に示す。

表 3-1 非機能要件一覧

カテゴリ	ID	非機能項目	要件詳細
可用性	NF001	業務の継続性	● 実証期間中に対処可能な障害が発生した場合、業務再開までベストエフォートでの対応を行うものとすること。この時、業務再開までの停止許容時間、目標復旧時間は設けないこと。AWS 基盤障害など、クラウド事業者側で復旧対応が必要となる障害は対処可能な障害に含まないものとすること。
	NF002	データのバックアップ・リカバリ	● AWS で提供されるサービス、機能を用いて日次バックアップを取得する。バックアップ対象に応じて、オフラインバックアップとオンラインバックアップを組み合わせて取得すること。また、障害発生時などデータ復旧を必要とする場合は、取得したバックアップデータから同サービス、機能を用いて復旧すること。
性能・拡張性	NF003	スマホアプリケーションの操作レスポンス	● 診察予約後にデマンドバス配車が完了するまでの処理時間が想定シナリオ実行に影響を与えない範囲となっているか。 ● 実証実施時点では実績を記録し、どの程度の値が望ましいか又は許容可能かをモデル仕様書に記載する。 ● システム連携の応答時間については、電子カルテシステムで入力し診察予約が Gateway に連携（5 分毎）され次第、10 秒以内に PHR システムとデマンドバス配車システムに連携する。
	NF004	スケーラビリティと拡張性	● 各システムリソースの拡張を容易に行うことができるシステム構成を採用すること。
運用・保守性	NF005	外部システム接続	● 既存システムへの影響を回避するため、既存機能への変更は局所化し、両接続仕様を確認するための外部連携リハーサルを計画すること。
セキュリティ	NF006	データの秘匿	● 機密性のあるデータの取扱いは極力最低限とし、必要な際は伝送時や蓄積時に秘匿するための暗号化すること。
システム環境・エコロジー	NF007	システム制約/前提条件	● 機密情報や個人情報等を取り扱う場合、情報利用者やアクセス方法の整備を行うこと。

3-2. 非機能要件の詳細

【NF001】業務の継続性

- 要件詳細
  - 実証期間中に対処可能な障害が発生した場合、業務再開までベストエフォートでの対応を行うものとすること。この時、業務再開までの停止許容時間、目標復旧時間は設けないこと。AWS 基盤障害など、クラウド事業者側で復旧対応が必要となる障害は対処可能な障害に含まないものとすること。
- 本非機能要件の対象となる機能ブロック
  - 診察予約
  - 離院時刻予測
  - ユーザー向け機能
  - デマンドバス配車システムとの連携
  - 基本機能
  - 配車情報管理
  - 病院システムとの連携（配車予約の生成）
  - 病院システムとの連携（配車時刻の取得）
  - 予約情報連携
  - 立ち寄り先案内
- 設定理由
  - 障害発生時においても実証実験が継続できるよう、復旧方式、手順を確立する一方、用途やユーザーが限定的であるため、業務再開までの明確な目標復旧時間は設けず、ベストエフォートでの対応を行うものとする。
- 評価方法
  - 本システムにおける障害発生時の復旧方式を設計し、設計内容に基づく障害発生時を想定した復旧確認を行う。

【NF002】データのバックアップ・リカバリ

- 要件詳細
  - AWS で提供されるサービス、機能を用いて日次バックアップを取得する。バックアップ対象に応じて、オフラインバックアップとオンラインバックアップを組み合わせて取得する。また、障害発生時などデータ復旧を必要とする場合は、取得したバックアップデータから同サービス、機能を用いて復旧する。
- 本非機能要件の対象となる機能ブロック
  - 診察予約
  - 離院時刻予測
  - 基本機能
  - 配車情報管理
  - 病院システムとの連携（配車予約の生成）

- 病院システムとの連携（配車時刻の取得）
- 予約情報連携
- 立ち寄り先案内

- 設定理由
  - データ損失を伴う障害が発生した場合にも実証実験が継続できるよう、定期的なバックアップデータの取得を行う。また、可能な限り障害発生時直前の状態に復元ができるよう、バックアップ頻度は日次とする。また、より簡易な手順でバックアップ取得及び障害発生時の復元が可能となるよう、AWS で提供されるサービス、機能を用いたバックアップ方式を採用する。
- 評価方法
  - 本システムにおけるバックアップ方式を設計し、設計内容に基づく障害発生時を想定したバックアップ、リカバリ確認を行う。

【NF003】スマホアプリケーションの操作レスポンス

- 要件詳細
  - 診察予約後にデマンドバス配車が完了するまでの処理時間が想定シナリオ実行に影響を与えない範囲となっているか。
- 本非機能要件の対象となる機能ブロック
  - 診察予約
  - 離院時刻予測
  - ユーザー向け機能
  - デマンドバス配車システムとの連携
  - 基本機能
  - 配車情報管理
  - 病院システムとの連携（配車予約の生成）
  - 病院システムとの連携（配車時刻の取得）
  - 予約情報連携
  - 立ち寄り先案内
- 設定理由
  - 操作レスポンスが実業務の妨げにならないかの確認のため。
- 評価方法
  - システムテスト結果を踏まえ、実証シナリオ作成時に具体的な目標範囲値設定、実証実施時点では実績を記録し、どの程度の値が望ましいか又は許容可能かをモデル仕様書に記載する。

【NF004】スケーラビリティと拡張性

- 要件詳細
  - 各システムリソースの拡張を容易に行うことができるシステム構成を採用する。
- 本非機能要件の対象となる機能ブロック
  - デマンドバス配車システムとの連携
  - 病院システムとの連携（配車予約の生成）
  - 病院システムとの連携（配車時刻の取得）
  - 予約情報連携
  - 立ち寄り先案内
- 設定理由
  - 今後の展開を見据え、将来のデータ量の増加や新機能の追加に対応できるよう、必要に応じた各システムリソースの拡張を容易に行うことができるシステム構成を採用する。
- 評価方法
  - 本システムを構成するシステムリソースについて、拡張方針および拡張方式を設計し、実際に設計内容に基づく拡張が可能であることを操作確認する。

【NF005】外部システム接続

- 要件詳細
  - 既存システムへの影響を回避するため、既存機能への変更は局所化し、両接続仕様を確認するための外部連携リハーサルを計画すること。
- 本非機能要件の対象となる機能ブロック
  - デマンドバス配車システムとの連携
  - 病院システムとの連携（配車予約の生成）
  - 病院システムとの連携（配車時刻の取得）
  - 予約情報連携
  - 立ち寄り先案内
- 設定理由
  - 稼働中の業務への影響を回避するため。
- 評価方法
  - 設計工程における既存影響点検と、テスト工程における接続リハーサルを行う。

【NF006】データの秘匿

- 要件詳細
  - 機密性のあるデータの取扱いは極力最低限とし、必要な際は伝送時や蓄積時に秘匿するための暗号化すること。
- 本非機能要件の対象となる機能ブロック
  - 診察予約
  - 離院時刻予測

- 基本機能
- 配車情報管理
- 病院システムとの連携（配車予約の生成）
- 病院システムとの連携（配車時刻の取得）
- 予約情報連携
- 立ち寄り先案内

- 設定理由
  - 病院にて保管される患者が特定される情報の漏洩を回避するため。
- 評価方法
  - 設計工程において連携に必要なデータは最小限とされていることをレビューにて確認する。

【NF007】システム制約/前提条件

- 要件詳細
  - 機密情報や個人情報等を取り扱う場合、情報利用者やアクセス方法の整備を行うこと。
- 本非機能要件の対象となる機能ブロック
  - 診察予約
  - 離院時刻予測
  - 基本機能
  - 配車情報管理
  - 病院システムとの連携（配車予約の生成）
  - 病院システムとの連携（配車時刻の取得）
  - 予約情報連携
  - 立ち寄り先案内
- 設定理由
  - 病院にて保管される患者および秘匿情報の漏洩を回避するため。
- 評価方法
  - データアクセス時にデータ所有者とのルールの取り決め及び書面による合意がなされているかを確認する。

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

実証調査に利用するデータ一覧について以下のとおり示す。

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	デマンドバス利用者情報	CSV 形式	徳島県立中央病院 徳島県交通政策課 富士通	FN009
DT002	デマンドバス乗降場情報	CSV 形式	徳島県立中央病院 徳島県交通政策課	FN009
DT003	復路立ち寄り先データ	JSON 形式	徳島県交通政策課 徳島県立中央病院	FN008
DT004	診察予約・検査予約データ	XLSX 形式	徳島県立中央病院	FN003
DT005	診察・検査実績データ	XLSX 形式	徳島県立中央病院	FN003
DT006	診察進捗実績平均データ	CSV 形式	DT005	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】デマンドバス利用者情報

- 本データの概要
  - 本実証調査においてデマンドバスに乗車し診察を行う被験者に関する CSV 形式のデータである。
- データ定義

表 4-2 デマンドバス利用者情報のデータ定義

No.	日本語名称	項目長	必須	書式・選択肢など
1	利用者番号	-	○	桁数は提供者ごとに設定する。
2	姓	-	○	-
3	姓（ふりがな）	-	○	-
4	名	-	○	-
5	名（ふりがな）	-	○	-
6	生年月日	-	○	yyyy/m/d
7	性別	-	○	男 or 女 or 無回答
8	郵便番号	-	○	-
9	住所	-	○	-
10	電話番号 1	-	○	メイン番号
11	電話番号 2		-	サブ番号
12	メールアドレス	-	-	メール連絡「必要」の場合、必須
13	その他 1	-	-	障害者手帳保持（該当する場合のみ）
14	その他 2	-	-	要介助（該当する場合のみ）
15	その他 3	-	-	要車椅子（該当する場合のみ）
16	その他 4	-	-	免許返納（該当する場合のみ）
17	メモ	-	-	-
18	自宅緯度	-	-	緯度、経度データ登録ある場合、自宅乗降場を設定する。
19	自宅経度	-	-	-

20	自宅以外の最寄り乗降場番号	-	-	たとえば、「22_上野バス停」という乗降場の「22」を設定する。
21	自宅以外の最寄り乗降場名称	-	-	乗降場番号の補足情報（システム上は不要）
22	統計地区	-	-	自宅乗降場の場合のみ設定する。
23	運行区画	-	-	自宅乗降場の場合のみ設定する。
24	電話連絡	-	○	必要又は不要
25	メール配信（予約時）	-	○	必要又は不要、メールアドレス登録ある場合のみ利用可
26	メール配信（乗車時）	-	-	必要又は不要、オプション項目
27	メール配信（降車時）	-	-	必要又は不要、オプション項目
28	ユーザ種別	-	○	0 固定
29	オペレーターメモ	-	-	車載器に連携されない内部メモ
30	基本料金	-	-	

- データ形式
  - CSV 形式
- 出所
  - 徳島県立中央病院、徳島県交通政策課、富士通にて選定
- 変換方法
  - -

【DT002】デマンドバス乗降場情報

- 本データの概要
  - 本実証調査においてデマンドバスが発着を行う乗降場の CSV 形式のデータである。
- データ定義

表 4-3 デマンドバス乗降場情報のデータ定義

No.	日本語名称	項目長	必須	書式・選択肢など
1	大項目	-	○	検索時に使用する。
2	小項目	-	○	検索時に使用する。
3	統計地区	-	○	集計時に使用する。
4	No.	-	○	識別子
5	名称	-	○	正式名称
6	ふりがな	-	○	正式名称のふりがな
7	緯度	-	○	-

8	経度	-	○	-
9	住所	-	-	-
10	検索ワード	-	-	検索対象としたい別名等（半角スペースを使い複数登録可能）
11	メモ	-	-	-
12	行政区域	-	-	オペレーター画面表示の地区
13	運行区画	-	○	運行ルール制御用の地区

- データ形式
  - CSV 形式
- 出所
  - 被験者の住所や行動範囲を考慮し、徳島県立中央病院、徳島県交通政策課にて選定
- 変換方法
  - -

【DT003】復路立ち寄り先データ

- 本データの概要
  - 本実証調査において病院の復路における立ち寄り先（全利用者共通）が記載されたデータである。
- データ定義

```
{
    場所名：○○スーパー,
    内部コード：1
},
{
    場所名：○○薬局,
    内部コード：2
},
{
    場所名：△△薬局,
    内部コード：3
},
```

- データ形式
  - JSON 形式
- 出所
  - 実証参加者の属性を踏まえ、立ち寄り先候補について病院や県交通政策課からヒアリングする。ヒアリングした結果から立ち寄り先を 2～3 件決定し、立ち寄り先名称と内部コードを当データに書き込む。

- 変換方法
  - -

【DT004】診察予約・検査予約データ

- 本データの概要
  - 本実証調査において、徳島県立中央病院の外来診察予約情報や検査予定内容が記録された予約情報である。
- データ定義

表 4-4 診察予約データのデータ定義

No.	日本語名称	項目長	必須	書式・選択肢など
1	利用者番号	-	○	数字
2	予約日	-	○	YYYYMMDD 形式
3	予約時間	-	○	HHMMSS 形式
4	予約枠コード	-	○	英数字
5	予約枠名称	-	○	日本語表記（最大 10 文字）名称

表 4-5 検査・処置予約（依頼）データのデータ定義

No.	日本語名称	項目長	必須	書式・選択肢など
1	利用者番号	-	○	数字
2	検査・処置日	-	○	YYYYMMDD 形式
3	検査・処置予定時間	-	○	HHMMSS 形式
4	検査・処置内容コード	-	○	英数字
5	検査・処置名称	-	○	日本語表記（最大 10 文字）名称

- データ形式
  - XLSX 形式
- 出所
    - モニター対象利用者の診察関連の情報につき徳島県立中央病院から入手（電子カルテデータベースより対象データ項目を抽出）
- 変換方法
  - -

【DT005】診察・検査実績データ

- 本データの概要
  - 本実証調査において、徳島県立中央病院の泌尿器科における外来診察情報や検査実施内容が記録された実績情報である。

- データ定義

表 4-6 診察データのデータ定義

No.	日本語名称	項目長	必須	書式・選択肢など
1	利用者番号	-	○	数字
2	受付日	-	○	YYYYMMDD 形式
3	受付時間	-	○	HHMMSS 形式
4	予約枠コード	-	○	英数字
5	予約枠名称	-	○	日本語表記（最大 10 文字）名称

表 4-7 検査・処置実施データのデータ定義

No.	日本語名称	項目長	必須	書式・選択肢など
1	利用者番号	-	○	数字
2	検査・処置日	-	○	YYYYMMDD 形式
3	検査・処置時間	-	○	HHMMSS 形式
4	検査・処置内容コード	-	○	英数字
5	検査・処置名称	-	○	日本語表記（最大 10 文字）名称

- データ形式
  - XLSX 形式
- 出所
  - モニター対象利用者の診察関連の情報につき徳島県立中央病院から入手（電子カルテデータベースより対象データ項目を抽出）
- 変換方法
  - -

【DT006】診察進捗実績平均データ

- 本データの概要
  - 本実証調査において、徳島県立中央病院で実施された診察、検査、処置、会計についての実績時間である。
- データ定義

表 4-8 診察進捗実績平均データのデータ定義

No.	日本語名称	項目長	必須	書式・選択肢など
1	診療科コード	-	○	英数字
2	診察平均時間	-	○	HHMMSS 形式
3	検査平均時間	-	○	HHMMSS 形式
4	処置平均時間	-	○	HHMMSS 形式
5	会計平均時間	-	○	HHMMSS 形式

- データ形式
  - CSV 形式
- 出所
  - 泌尿器科の診察関連の情報につき徳島県立中央病院から入手した【DT005】診察・検査実績データを加工する。
- 変換方法
  - 処理内容・手順
    - 電子カルテのデータベースより対象データ項目を抽出し出力する。
    - 出力データについて実績として採用されている項目について平均値を算出する。
  - 利用したソフトウェア
    - Excel

5. 用語集

用語	定義・説明
患者	● 本実証に関わらず一般的な患者を指す言葉
患者 ID	● 電子カルテシステムで利用する患者の一意キー
交通利用者	● 通院に伴いデマンドバスを利用される方
デマンドバス配車システム	● デマンドバスの配車を管理するシステムであり、利用者アプリ、ドライバーアプリ、オペレーターアプリ、レポーティングアプリを含む ● 予約から配車までデマンドバスに関わるすべての機能を提供する
電子カルテシステム	● 病院内で利用される電子カルテ（患者情報管理）システム
被験者	● 本実証実験に限った利用者
PHR	● Personal Health Record の略であり、PHR アプリと表現する場合はスマホ上で動作する患者向けヘルスケア情報アプリケーションを指す。PHR システムと表現する場合は、PHR アプリやヘルスケア MaaS Gateway を動作させるための機能がクラウド上に配備されたシステムを指す。
ヘルスケア MaaS Gateway	● 病院の診療終了時刻を予測する「離院時刻予測モデル」と PHR システムとデマンドバス配車システムの連携インターフェース ● ヘルスケア MaaS Gateway ライブラリ表現する場合は連携インターフェースで上を構成するファイル群を指す
利用者	● 本実証に関わらず将来的な観点含め当サービスの機器やサービスを享受する者
利用者番号	● デマンドバス配車システムで利用する利用者の一意キー

COMmmONS by MLIT

ヘルスケアMaaSシステム 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:　富士通株式会社