

# パーソナルデータ連携モジュール

pxr-identity-verify-service ビルド手順書

2022 年 10 月 1 日

日本電気株式会社

## 改版履歴

版	作成日	変更内容
1.0	2022/10/1	新規作成

# 目次

<b>1</b>	<b>はじめに</b> .....	<b>4</b>
1.1	前提条件 .....	4
<b>2</b>	<b>ビルド手順</b> .....	<b>5</b>
2.1	サービスをビルドする .....	5
<b>3</b>	<b>Unit Test 手順</b> .....	<b>6</b>
3.1	DB を作成する（1 環境につき初回のみ） .....	6
3.2	SCHEMA, TABLE を作成する（1 環境につき初回のみ） .....	6
3.3	UNIT TEST を実行する .....	7
<b>4</b>	<b>pxr-identity-verify-service 起動手順</b> .....	<b>8</b>
4.1	PXR-IDENTITY-VERIFY-SERVICE を起動する .....	8
4.2	WEB ブラウザでアクセスする .....	8
<b>5</b>	<b>Docker コンテナイメージ作成手順</b> .....	<b>9</b>
5.1	DOCKER コンテナイメージを作成する .....	9
5.2	DOCKER コンテナイメージをレジストリに登録する .....	9

# 1 はじめに

本書は、パーソナルデータ連携モジュールの一部である、pxr-identity-verify-service のビルド手順および Unit Test 手順について記載・説明する。

## 1.1 前提条件

- Node (12.22.10) がインストールされていること
  - PostgreSQL (12.x) がインストールされていること
  - Docker (20.x) がインストールされていること
- ※ Docker コンテナを使用したパーソナルデータ連携モジュールを構築する場合

## 2 ビルド手順

pxr-identity-verify-service のビルド手順について記載する。

※本書では作業ディレクトリをホームディレクトリ配下としているが、任意のディレクトリを作業ディレクトリとすることも可能である(その場合は作業ディレクトリを読み替えて実行すること)。

### 2.1 サービスをビルドする

事前準備として、作業ディレクトリ配下に「pxr-identity-verify-service」のプロジェクトを配置しておくこと。  
以下のコマンドを実行し、エラーが出ないことを確認する。

Linux	Windows (PowerShell)
\$ cd ~/pxr-identity-verify-service \$ npm i \$ npm run build	\$ cd ~/pxr-identity-verify-service \$ npm i \$ npm run build

## 3 Unit Test 手順

pxr-identity-verify-service の Unit Test 手順について記載する。

### 3.1 DB を作成する（1 環境につき初回のみ）

以下を実行する。

（Linux 環境はコマンドラインで実行した例を、Windows 環境では pgAdmin4 を利用した例を示す）

Linux	Windows
<pre>\$ psql -U postgres ---- postgres=# CREATE DATABASE pxr_pod         WITH         OWNER = postgres         ENCODING = 'UTF8'         LC_COLLATE = 'C'         LC_CTYPE = 'C'         TABLESPACE = pg_default         CONNECTION LIMIT = -1 ; ----</pre>	<ul style="list-style-type: none"><li>・pgAdmin4 を起動する</li><li>・左のメニューから Servers&gt; PostgreSQL 12&gt; データベースの順に開き、データベースを右クリックして作成&gt; データベースを選択する</li><li>・データベースに「pxr_pod」と入力して保存する</li></ul>

### 3.2 Schema, Table を作成する（1 環境につき初回のみ）

事前準備として、作業ディレクトリ配下に ddl ディレクトリを配置しておくこと。

以下を実行する。

（Linux 環境はコマンドラインで実行した例を、Windows 環境では pgAdmin4 を利用した例を示す）

Linux	Windows
<pre>\$ cd ~/ddl/db/pxr-identity-verify-service \$ psql -U postgres -d pxr_pod -f createDB.sql \$ psql -U postgres -d pxr_pod -f createTable.sql</pre>	<ul style="list-style-type: none"><li>・3 で作成した pxr_pod を右クリックして、クエリツールを選択する</li><li>・右側に表示された画面で、ファイルを開くを選択し、ddl リポジトリの db¥pxr-identity-verify-service 配下にある createDB.sql を開く</li><li>・実行を選択し、「ログイン/グループロール」に pxr_identify_verify_user が作成されていること、pxr_pod のスキーマ配下に</li></ul>

	pxr_identify_verify が作成されていることを確認する ・クエリツール画面で、ddl リポジトリの db¥pxr-identity-verify-service 配下にある createTable.sql を開いて、実行する
--	---

### 3.3 Unit Test を実行する

以下のコマンドを実行し、エラーが出ないことを確認する。

Linux	Windows (PowerShell)
<pre>\$ cd ~/pxr-identity-verify-service \$ npm run jest-clear \$ npm run test:unit</pre>	<pre>\$ cd ~/pxr-identity-verify-service \$ npm run jest-clear \$ npm run test:unit</pre>

## 4 pxr-identity-verify-service 起動手順

pxr-identity-verify-service の起動手順について記載する。

### 4.1 pxr-identity-verify-service を起動する

以下のコマンドを実行する。

Linux	Windows (PowerShell)
\$ cd ~/pxr-identity-verify-service \$ npm run start	\$ cd ~/pxr-identity-verify-service \$ npm run start

### 4.2 Web ブラウザでアクセスする

以下を実行する。

Linux	Windows
Web ブラウザで以下にアクセスし、Swagger が表示されること <a href="http://localhost:3007/api-docs/">http://localhost:3007/api-docs/</a>	Web ブラウザで以下にアクセスし、Swagger が表示されること <a href="http://localhost:3007/api-docs/">http://localhost:3007/api-docs/</a>



## 5 Docker コンテナイメージ作成手順

Docker コンテナイメージを作成する手順について記載する。

コンテナを使用したパーソナルデータ連携モジュールの構築手順については以下を参照すること。

パーソナルデータ連携モジュール 構築ガイド

### 5.1 Docker コンテナイメージを作成する

以下のコマンドを実行する。

Linux	Windows (PowerShell)
\$ cd ~/pxr-identity-verify-service \$ docker build -t {イメージ名}:{タグ} .	\$ cd ~/pxr-identity-verify-service \$ docker build -t {イメージ名}:{タグ} .

### 5.2 Docker コンテナイメージをレジストリに登録する

以下のコマンドを実行する。

Linux	Windows (PowerShell)
\$ cd ~/pxr-identity-verify-service \$ docker tag {イメージ名}:{タグ} {Docker リポジトリ名}/{イメージ名}:{タグ} \$ docker push {Docker リポジトリ名}/{イ メージ名}:{タグ}	\$ cd ~/pxr-identity-verify-service \$ docker tag {イメージ名}:{タグ} {Docker レジストリ名}/{イメージ名}:{タグ} \$ docker push {Docker レジストリ名}/{イ メージ名}:{タグ}