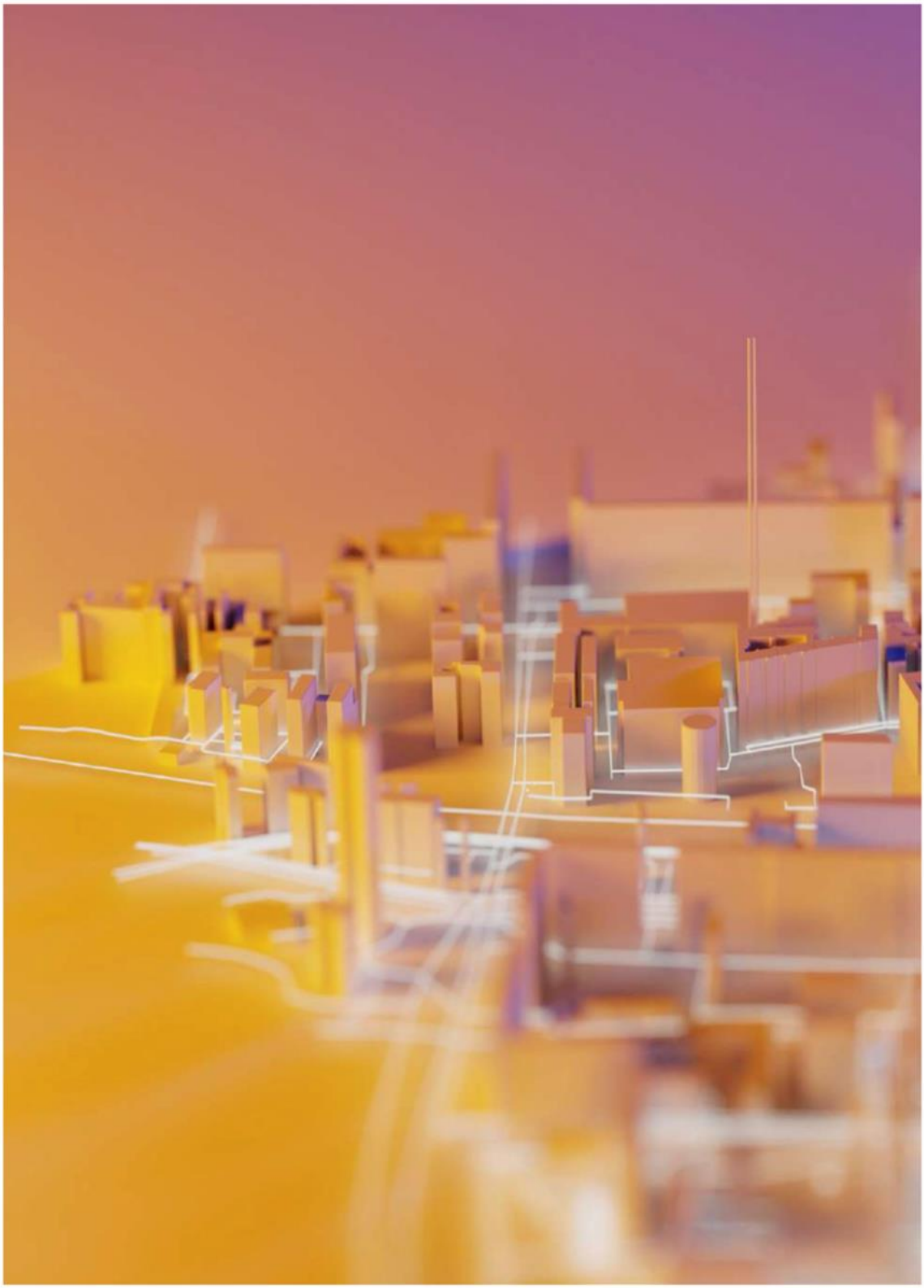
PLATEAU
by MLIT

Handbook of 3D City Models

3D都市モデル導入のためのガイドブック



PLATEAU VIEW 構築マニュアル

series No. 09

PLATEAU VIEW Setup Manual

はじめに

- Project PLATEAUでは、 2020年度に3D都市モデルのPLATEAU VIEW構築のための実証調査を実施した。本実証調査は、3D都市モデル及びこれを活用したユースケース開発のための可視化環境である「PLATEAU VIEW」を開発することで、3D都市モデルがもたらすソリューションの価値を検証することを目的としている。
- 2021年度には、クリッピング機能や日射シミュレーション機能など、Web上で実施可能な解析機能を追加「PLATEAU VIEW 1.1」をリリースした。また、「第3章　PLATEAU VIEWの構築手順」の内容を大幅に拡充し、PLATEAU VIEWを構築するためのチュートリアルを充実させるとともに、Project PLATEAU GitHub上でTerria.js用カタログ生成アプリとそのチュートリアルを公開した。
- 2022年度には、データセットの可視化機能に限定されていた「PLATEAU VIEW 1.1」を発展させ、データ登録・管理・配信機能及び機能追加を行った「PLATEAU VIEW 2.0」を開発し、GitHub上でソーススクリプトを公開した。さらに、PLATEAU VIEWを利用するための技術チュートリアルを公式ウェブサイト上で公開した。
- 2023年度には、「PLATEAU VIEW 2.0」をUI/UXの観点からの見直しと、3DCG技術を利用した描画品質の向上を図るとともに、PLATEAU CMSのデータ配信APIやコンテンツ管理機能の改善を行った「PLATEAU VIEW 3.0」を開発した。
- 2024年度には、空間IDの可視化及び連携、メッシュ単位でのCityGMLデータ配信のほか、データの変換及び演算処理を行うシステムPLATEAU Flowを実装した「PLATEAU VIEW 4.0」を開発した。
- PLATEAU VIEWの構築を検討するに当たっては、Project PLATEAUのオープンデータの思想に基づき、できるだけオープンソースのフレームワークを利用することで、ベンダーロックを回避する形でシステム構成を行うことを心掛けた。また、オープンソースの利用には、自治体等が低コストで類似のシステムを構築できることや、技術者コミュニティとの連携によるシステムの持続的な発展が期待されること等のメリットもある。
- 本マニュアルは、PLATEAU VIEWの開発により得られた成果をもとに、その機能、システム環境、仕様、構築手法等を解説することで、自治体や民間企業、技術者コミュニティ等に所属する多様なプレイヤーが3D都市モデルの可視化環境を構築する際に参照できる知見を提供し、3D都市モデルの整備及びこれを活用したユースケース開発への参画のすそ野を広げることを目的とするものである。
- 自治体や民間企業、技術者等の多くの方に本マニュアルを参照していただき、Project PLATEAUの技術コミュニティがさらに発展することを期待する。

アップデートノート

2025.3.21　「PLATEAU VIEW構築マニュアル ver5.0」

2024年度の調査結果をふまえ、以下の項目を改訂した。

- PLATEAU VIEWの機能追加、PLATEAU Flowを追加したことに伴い、全体の構成を見直し、「第1章 PLATEAU VIEW 4.0の構成」、「第２章　PLATEAU VIEW 4.0の機能」、「第３章 PLATEAU VIEW 4.0の環境構築」とアップデートした。

2024.3.22　「PLATEAU VIEW構築マニュアル（旧実証環境構築マニュアル）ver4.0」

2023年度の調査結果をふまえ、以下の項目を改訂した。

- 「実証環境構築マニュアル」を「PLATEAU VIEW構築マニュアル」と改称し、2023年度における最新仕様のみをマニュアルに反映した。
- 「第3章　PLATEAU VIEWの構築手順」にAWS向けセットアップや、WebAR向け構築手順を追加。

2023.3.22　「実証環境構築マニュアル ver3.0」

2022年度の調査結果をふまえ、以下の項目を改訂した。

- 「第1編　PLATEAU VIEW 2.0」を新たにリリースしたPLATEAU VIEW 2.0編として拡充
- 「第2編　PLATEAU VIEW 1.1」をPLATEAU VIEW 1.1編として改訂

2022.3.25　「実証環境構築マニュアル ver2.0」

2021年度の調査結果をふまえ、以下の項目を改訂した。

- 「1.4　ソフトウェア構成」を新たにリリースしたPLATEAU VIEW 1.1の構成にアップデート
- 「第2章　実証環境の構築手順」の内容を大幅に拡充
- 「第3章 ユーザーマニュアル」をPLATEAU VIEW 1.1に合わせて改訂

2021.3.26　「実証環境構築マニュアル ver1.0」

■目次

用語集

本マニュアルの目的

第1章　PLATEAU VIEW 4.0の構成 ・・・・・・・・・・ **9**

1.1　システム構成 ・・・・・・・・・・ 10

1.1.1 PLATEAU VIEW 4.0

1.1.2 PLATEAU CMS

1.1.3 PLATEAU Editor / VIEW

1.1.4 PLATEAU Flow

1.2　PLATEAU VIEW 4.0の改善点 ・・・・・・・・・・ 21

1.2.1 空間IDから属性情報を取得する機能

1.2.2 地域メッシュからCityGMLをダウンロードする機能

1.2.3 CityGML API

第２章　PLATEAU VIEW 4.0の機能 ・・・・・・・・・・ **30**

2.1　PLATEAU CMS ・・・・・・・・・・ 31

2.1.1 PLATEAU CMSとは

2.1.2 管理者向け機能

2.1.3 データ登録者向け機能

2.1.4 API仕様

2.2　PLATEAU Editor / VIEW ・・・・・・・・・・ 60

2.2.1 PLATEAU Editor / VIEWとは

2.2.2 PLATEAU Editor の機能

2.2.3 PLATEAU VIEW の機能

2.3　PLATEAU Flow ・・・・・・・・・・ 95

2.3.1 PLATEAU Flowとは

2.3.2 ワークフロー構築・実行方法

2.3.3 その他の機能

2.3.4 簡単なワークフローを構築して実行する

第３章　PLATEAU VIEW 4.0の環境構築 ・・・・・・・・・・ **134**

3.1　PLATEAU CMS / Editor / Flowの環境構築 ・・・・・・・・・・ 135

3.1.1 システム構成の全体像

3.1.2 各種サービスのセットアップ

3.1.3 Google Cloud 向けセットアップ

3.1.4 PLATEAU CMSの動作確認・セットアップ

3.1.5 PLATEAU Editorの動作確認

3.1.6 PLATEAU Flowの動作確認

3.2　PLATEAU Editor / VIEWの設定 ・・・・・・・・・・ 146

3.2.1 プロジェクトの作成

3.2.2 シーンの設定

3.2.3 ウィジェットの設定

3.3　FME Flow ・・・・・・・・・・ 150

3.3.1 FME Flowとその稼働環境

3.3.2 FME Flow プロジェクトファイル

3.3.3 環境の準備

3.3.4 構築の手順

3.3.5 FME Flowの設定

付録　テライン配信の整備 ・・・・・・・・・・ **156**

1.1　Cesium向けの地形データ配信 ・・・・・・・・・・ 158

1.2　Mapbox GL JS・MapLibre GL JS向けのテライン配信 ・・・・・・・・ 164

用語集

PLATEAU VIEW 4.0において独自に使用される用語をまとめる。

表　用語集：ユーザー

用語	説明
国交省職員・システム管理者	PLATEAU CMS上での登録データのレビュー、PLATEAU Editor上での公開ページの編集等を行う。
地方自治体	PLATEAU CMSおよびEditorを利用して独自のVIEW構築を行う。
データ制作事業者	PLATEAU CMS上でPLATEAU関連データセットの登録を行う。
データ利用者	G空間情報センターへ登録されたPLATEAUデータを利用する。
一般利用者	PLATEAU VIEWを利用し、PLATEAU関連データセットの閲覧を行う。
アプリ開発者	Unity/Unreal向けSDKを利用し、PLATEAUデータを利用したアプリケーション開発を行う。
JV	本実証環境構築事業においてシステムの開発および運用を担当するJV。

表　用語集：システム

用語	説明
PLATEAU CMS	PLATEAU関連データセットを管理する。
PLATEAU Editor	PLATEAU VIEW・自治体独自のビューアを編集・公開する。
PLATEAU VIEW	PLATEAU関連データセットを可視化する。
PLATEAU Flow	PLATEAUデータの品質検査・データ変換等を行う。
FMEサーバー	PLATEAUデータの品質検査・データ変換等を行う。
PLATEAU SDK for Unity/Unreal	Unity・Unreal Engine向けSDK。PLATEAUデータをゲームエンジン上で描画する。
G空間情報センター	PLATEAUデータをオープンデータとして公開する。

表　用語集：データ種別

用語	説明
PLATEAUデータ	都市局が定める「３D都市モデル標準製品仕様書」に準拠して作成された3D都市モデルのデータ。CityGML2.0形式で作成される。
ユースケースデータ	3D都市モデルデータに重畳して表示することを目的として作成されたユースケースに関するデータ。3D浸水想定区域図や動的データ、シミュレーションデータなど。
その他データ	PLATEAU VIEWで閲覧可能な、PLATEAUデータ及びユースケースデータ以外のデータ。避難施設、ランドマーク、行政界など。
オルソ画像	PLATEAUデータ整備に際して一部の都市で作成される空中写真データ。
PLATEAU関連データセット	PLATEAUデータ、ユースケースデータ、その他データセットの総称。

表　用語集：PLATEAU CMS

用語	意味
アカウント	管理者、自治体・受託事業者、ユースケースデータの登録者それぞれに 1 つずつ与えられるユーザーアカウントを指す。
ワークスペース	複数のユーザーが同じワークスペースで作業できる場所を指す。PLATEAU VIEW 4.0のワークスペースでは、管理者、自治体・受託事業者、ユースケースデータの登録者が全員同じワークスペース「PLATEAU」で操作する。
プロジェクト	ワークスペースには複数作成可能で、データ管理の目的に応じて作成する。管理者、自治体・受託事業者、ユースケースデータの登録者が全員同じプロジェクトで操作する。
モデル	データを管理する単位で、管理者のみが、管理するデータのスキーマを設定できる。プロジェクトの複数作成が可能。
スキーマ	モデルのデータ構造を指す。どんなフィールドがどんな型のデータを持つかや、制約条件などを定義する。管理者のみが定義可能で、自治体・受託事業者、ユースケースデータの登録者はこのスキーマに沿ってデータを入力する。
アイテム	管理するデータの最小単位で、スキーマに沿って実際のデータをアイテムとして登録する。3D都市モデルデータ登録者、ユースケースデータの登録者は対象データをアイテムとして入力する。
フィールド	管理するデータの属性を指す。整数値、文字列などのデータ型を管理者のみが設定する。
アセット	アップロードされたファイルを指す。サイズ・作成日時・URL・種類などのメタデータを持つ。複数ファイルの集まりをまとめて1つのアセットとして扱うことができる。管理者、自治体・受託事業者、ユースケースデータの登録者はPLATEAU関連データセットをアセットとしてアップロードする。
公開API	外部からアクセスされるPLATEAU CMSの認証不要のAPIを指す。登録したアイテムを公開リクエストで承認されるとそのアイテムが公開APIから配信されるようになる。PLATEAU VIEW 4.0 ではこの公開APIから配信されたデータがPLATEAU VIEWで使用され、自動的にデータカタログに掲載される。
公開リクエスト	登録したアイテムはすぐにはAPIとして配信されない。これを公開するためには管理者の承認を得ることが必要である。PLATEAU CMSでは、作成・更新したアイテムに対して、PLATEAU CMS上で公開の申請を出すことができる。この申請が承認されるとそのアイテムのデータが公開され、公開API経由で配信される。アイテムごとに個別に公開状況が管理されており、それぞれのアイテムごとに公開リクエストが必要である。
インテグレーション	PLATEAU CMS以外の外部アプリケーションとの連携機能を指す。PLATEAU CMSのワークスペースにインストールすることで、インテグレーションAPIを利用してデータの取得や変更を行ったり、任意のイベント発生時にWebhookを利用して外部アプリケーションと連携を行うことができる。

表　用語集：PLATEAU Flow

用語	意味
ワークスペース	Re:Earthなどの概念と同じ。メンバーに招待が必要な共同編集に必要なスペースの単位
プロジェクト	作業単位。複数のワークフローを含む。
ワークフロー	一連の処理の定義、処理の流れを示すもの。それぞれ実行可能。
ラン	ワークフローの実行状況。ワークフローを十個すると作成され、実行パラメーターやログなどの情報が紐づく。
アクション	リーダー・トランスフォーマー・ライターの総称。ワークフロー内に配置して接続できる。処理の最小単位。
リーダー（ソース）	ファイルなどからデータを読み込むアクション。
トランスフォーマー（プロセッサー）	データを受け取り何らかの処理を行い結果をデータとして出力するアクション。
ライター（シンク）	ファイルなどのデータに書き込むアクション。
データフレーム	アクション間を流れるデータの総称。レコードの集合体。
レコード	データフレーム内の1つ1つの「行」。
フィーチャー	レコードにジオメトリが存在する場合はフィーチャーと呼ぶ。
プロパティ	アクションのパラメータ。

本マニュアルの目的

PLATEAU VIEWとは、3D都市モデル及びこれを活用したユースケース開発のために可視化環境を提供するプログラム、サーバー、データ等の一連のシステムをいう。具体的な機能としては、3D都市モデルそれ自体を可視化することに加え、3D都市モデルと共に分析やシミュレーション等に用いられる各種データの可視化も行う。これにより、3D都市モデルの提供価値を検証することができる。

2020年度のProject PLATEAUでは、ウェブ上で閲覧可能なViewer「PLATEAU VIEW 1.0」を開発し、ウェブサイト「PLATEAU」上で公開した。続く、2021年度には、PLATEAU VIEW 1.0に機能追加を行った「PLATEAU VIEW 1.1」を開発し、アップデートを行った。

2022年度では、PLATEAU VIEW 1.1を発展させ、データ登録・管理・配信機能及び機能追加を行った「PLATEAU VIEW 2.0」を開発し、アップデートを行った。

2023年度には、「PLATEAU VIEW 2.0」をUI/UXの観点からの見直しと、3DCG技術を利用し、「PLATEAU VIEW 3.0」を開発し、アップデートを行った。

2024年度には、「PLATEAU VIEW 3.0」の機能を拡張し、データ検索・取得の効率化と利便性の向上を目的とした機能を導入し、「PLATEAU VIEW 4.0」を開発した。

PLATEAU VIEW 4.0は、データの検索・取得の効率化を図るため、「空間ID検索機能」や「メッシュコードを用いたCityGMLダウンロード機能」を導入した。また、地形配信サービスを拡充し、従来のCesiumJSだけでなく、Mapbox GL JSなどの他の地図エンジンでも利用できるようにした。さらに、データ変換・品質検査の効率化を目的とした「PLATEAU Flow」を開発し、ノーコードでのワークフロー作成やデータ変換を可能にした。

本マニュアルでは、PLATEAU VIEW 4.0の機能、システム環境、仕様、構築手法等を解説することで、自治体や民間企業等が3D都市モデルの可視化環境を構築する際に参照できる知見を提供し、3D都市モデルの整備及びこれを活用したユースケース開発を促進することを目的とするものである。

なお、 PLATEAU VIEW 4.0、 PLATEAU VIEW 3.0、PLATEAU VIEW 2.0及びPLATEAU VIEW 1.1のソースコードについては、Project PLATEAUのGitHubにおいてオープンソースとして公開しているので、参考にして頂きたい。

PLATEAU VIEW 4.0:
Project PLATEAU GitHub : https://github.com/Project-PLATEAU/PLATEAU-VIEW-4.0

PLATEAU VIEW 3.0:
Project PLATEAU GitHub : https://github.com/Project-PLATEAU/PLATEAU-VIEW-3.0

PLATEAU VIEW 2.0:
Project PLATEAU GitHub : https://github.com/Project-PLATEAU/PLATEAU-VIEW-2.0

PLATEAU VIEW 1.1:
Project PLATEAU GitHub : https://github.com/Project-PLATEAU/PLATEAU-VIEW-1.1

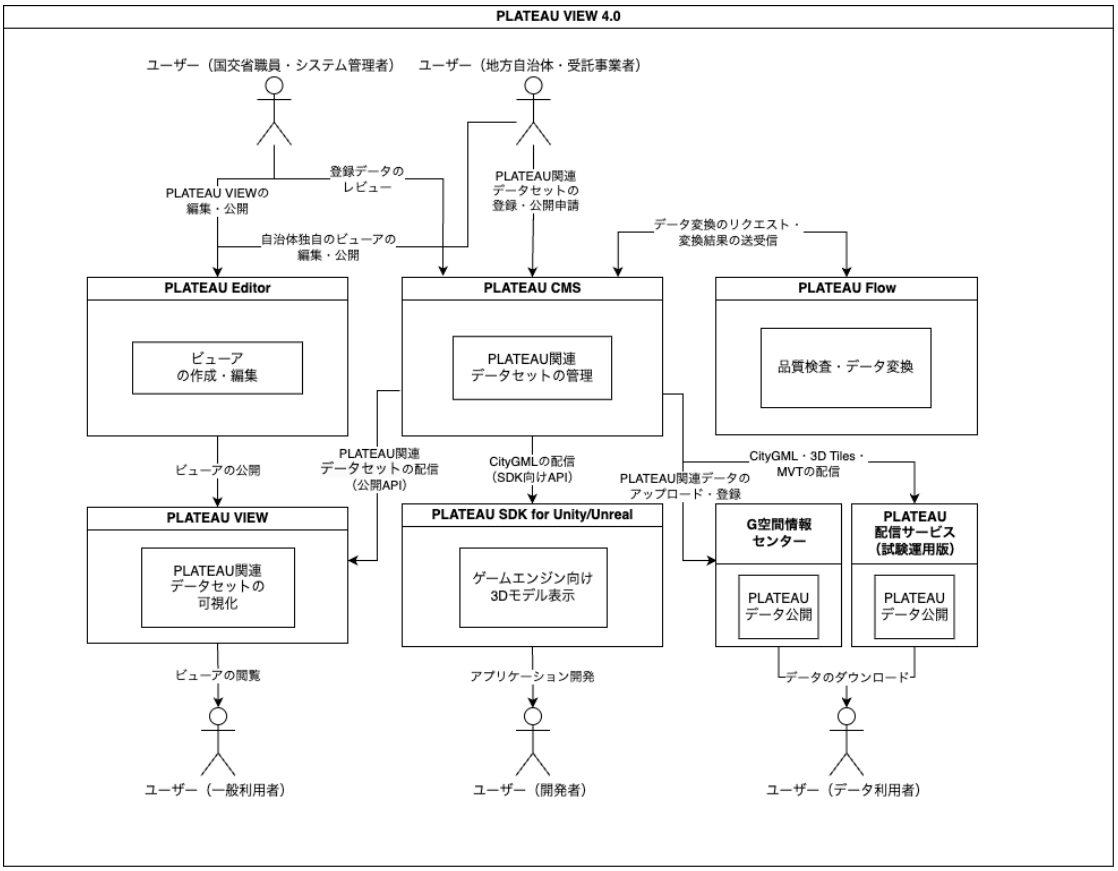
第１章　PLATEAU VIEW 4.0の構成

1.1 システム構成

1.1.1 PLATEAU VIEW 4.0

PLATEAU VIEW 4.0を構成するシステム及び想定ユーザーを解説する。PLATEAU VIEW 4.0は複数のシステムで構成されており、以下に全体図を示す。

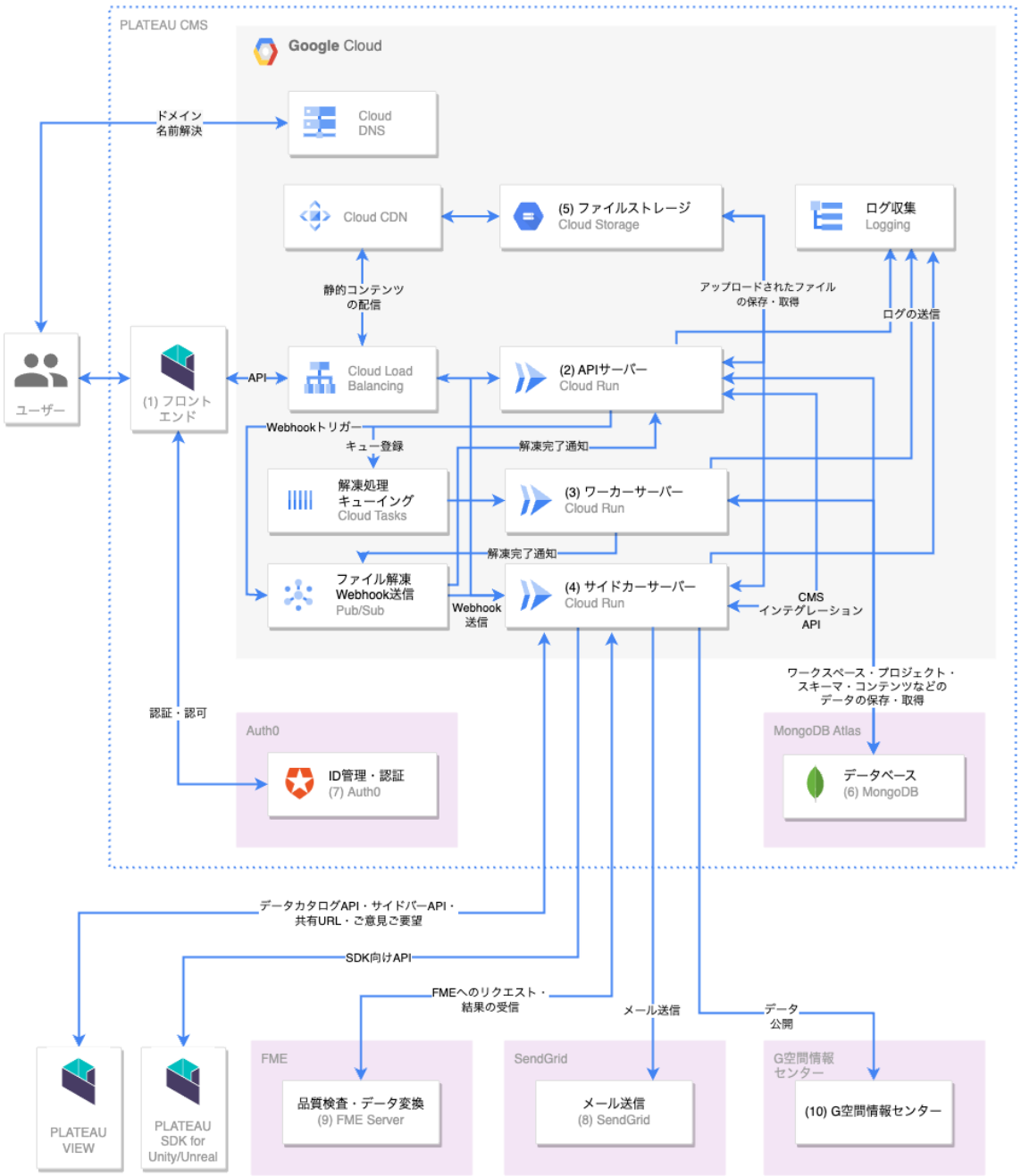
表　PLATEAU VIEW 4.0　システム全体構成



1.1.2 PLATEAU CMS

PLATEAU CMSのシステム構成について説明する。

表　PLATEAU CMSのシステム構成



（1）フロントエンド

Webブラウザ上で動作する、（HTML・CSS・JavaScriptによる）フロントエンドアプリケーション。APIサーバー・Auth0・その他各種サーバーとの通信を行い、UIやプレビュー機能による地図の表示を行う。

（2）APIサーバー（Cloud Run）

PLATEAU CMSのAPIサーバーであり、HTTPサーバーとして外部からのリクエストを受信している。MongoDBやGoogle Cloud Storageと連携して、アイテムの保存やプロジェクトの管理・公開などのさまざまなビジネスロジックを実行する。

APIサーバーは、Cloud Run 上で動作する。Cloud Run とは、Google Cloudで利用可能なサーバーレス（CaaS）プラットフォームであり、Dockerコンテナをデプロイすることで、サーバーの保守管理の手間なしに、アプリケーションをクラウド上で動作させることができる。同時接続リクエスト数が規定数以上に達すると自動的にコンテナが増加し、より多くのトラフィックを自動的に分散処理することができる。

Cloud Runは、デフォルト設定では、HTTPリクエストを受信して処理している間のみCPUが動作し、リクエストを処理していない時は動作を停止するため、HTTPリクエストを実際に受信し処理するために動作したCPU時間分のみが課金対象となる。この点が、常時稼働が前提となることが多いAWSのEC2やGoogle CloudのGoogle CloudEとは異なる。（2025年3月現在）PLATEAU VIEW 4.0のPLATEAU CMSのAPIサーバーは、メモリ4GB・CPU2コアの設定で動作している。

（3）ワーカーサーバー（Cloud Run）

PLATEAU CMSにおける非同期バックグラウンド処理を行うサーバーであり、HTTPサーバーとしてAPIサーバーからのリクエストを受信している。Zipファイル等の解凍処理やWebhookの送信などをCloud TasksやCloud PubSubと連携しながら行なっている。ワーカーサーバーは、APIサーバーと同じく、Cloud Run 上で動作する。（2025年3月現在）PLATEAU VIEW 4.0のPLATEAU CMSのWorkerサーバーは、メモリ32GB・CPU8コアの設定で動作している。

（4）サイドカーサーバー（Cloud Run）

PLATEAU CMSと連携しPLATEAU CMSを補助する形で動作するサーバー。外部サービスとの連携や、PLATEAU SDK for Unity/Unreal向けのデータ配信、PLATEAU Editor・PLATEAU VIEWで凡例等の表示を行うコンポーネント設定のデータ等を扱う。（コンポーネント設定については、2.2.2.3で詳細を解説）

- FMEへの品質検査・データ変換処理のリクエスト・結果の受信と保存
- G空間情報センターへのデータ登録処理・カタログ検査
- PLATEAU SDK for Unity/Unreal向けCityGMLデータ配信API
- PLATEAU VIEWの建築検索機能向け検索インデックスの構築
- その他データのランドマーク・鉄道駅・行政界データのCZMLへの変換
- PLATEAU Editor・PLATEAU VIEW向けAPI
    - データカタログAPI
    - サイドバー設定・共有URLデータの保存
    - ご意見ご要望を受け取りSendGridと連携してメール送信

サイドカーサーバーは、APIサーバーと同じく、Cloud Run 上で動作する。

（2025年3月現在）PLATEAU VIEW 4.0のPLATEAU CMSのPLATEAU VIEWサーバーは、メモリ16GB・CPU4コアの設定で動作している。

（5）ファイルストレージ（Google Cloud Storage）

フロントエンドのアプリケーションのソースコードや画像、ユーザーによってアップロードされたアセットファイルを保存する、オブジェクトストレージサーバー。

Google Cloud Storage（Google CloudS）を使用している。容量は無制限、自動的にスケーリングし、バックアップも自動的に行われ、データは複数拠点に分散配置される。巨大なファイルを格納・配信することが可能。

PLATEAU CMSでは、PLATEAU関連データセット等の静的ファイルをGoogle CloudSに保存している。

（6）MongoDB

MongoDBとは、ドキュメント指向のNoSQLデータベースで、データの柔軟性と拡張性が特徴。
PLATEAU CMSはデータベースとしてMongoDBを使用している。

PLATEAU CMSでは、MongoDB社が提供するクラウドサービス MongoDB Atlas を利用している。保守運用が自動化されたマネージドなMongoDBが利用可能で、自動的に3台以上のサーバーから成るクラスタを構成し、データベース内のデータは自動的に各サーバーに複製され、リクエストは分散処理されるようになっている。M0からM30まで、さまざまなマシンスペックのサーバーによる可用性の高いクラスタを構築することができ、マシンスペックによって料金が変わる。

（2025年3月現在）PLATEAU VIEW 4.0のPLATEAU CMS向けには、M10クラスタを運用しており、PLATEAU Editorと同じクラスタを使用している。
MongoDBには、ユーザーやワークスペース、スキーマ、コンテンツ（データカタログに表示される説明文等）に関する情報が保存される。

（7）Auth0

Auth0社が提供するクラウドサービス。アカウントの管理・認証・認可を行うIDプロバイダを提供する。Auth0を使用することで、開発者はアプリケーションに安全で使いやすく信頼性の高い認証認可機能を組み込むことができる。PLATEAU EditorもAuth0を使用して認証認可機能を実現している。なお、PLATEAU CMSとPLATEAU Editorでは同じAuth0テナントを利用している。

（2025年3月現在）テナントに対して登録されているユーザー数に応じて課金されるが、7000ユーザーまでは無料となっている。

（8）SendGrid

Twillio社が提供するクラウドサービス。クラウドベースのメール配信プラットフォームで、企業や開発者がアプリケーションやWebサイトから大量のメールを配信するために使用される。PLATEAU CMSではご意見ご要望のメール送信で使用している。

（9）FME Flow

Safe Software社（カナダ） が開発したデータ変換エンジンで、データの品質検査や、CityGMLから3D TilesやMapbox Vector Tilesへのデータ変換などを行う。詳細は「3.3 FME Flow」を参照されたい。

（10）G空間情報センター

官民問わずさまざまな主体により整備・提供される多様な地理空間情報を集約し、利用者がワンストップで検索・ダウンロードし利用できる、産学官の地理空間情報を扱うプラットフォーム。

PLATEAU CMSでは、3D都市モデルデータ、ユースケースデータの公開時（公開申請承認時）に自動的にG空間情報センターへ登録を行う。

（11）その他のコンポーネント

表　その他のコンポーネント

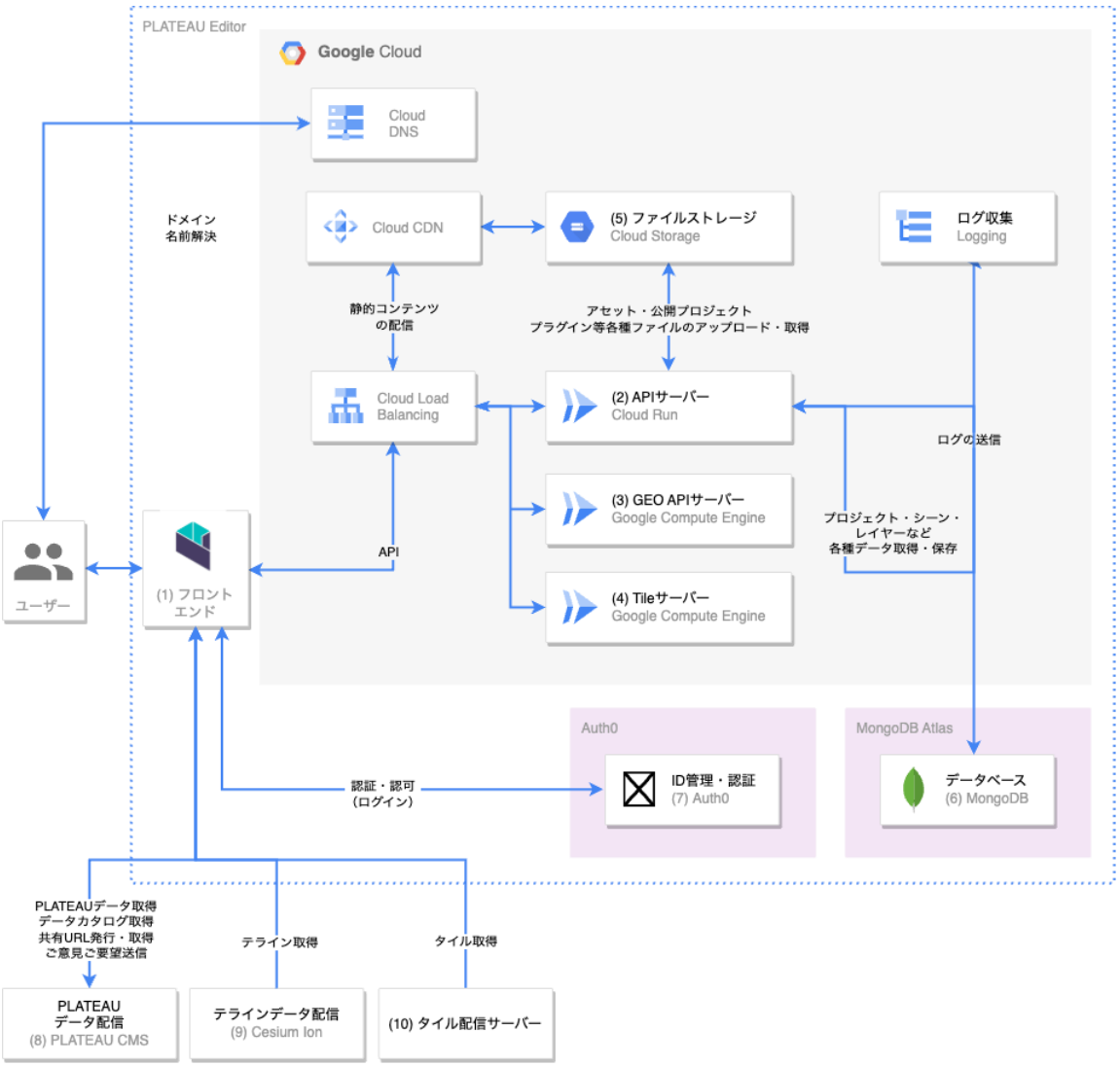
コンポーネント名	説明
Cloud CDN	Google Cloud(Google Cloud)のCDN（Content Deliver Network = ウェブコンテンツをインターネット経由で配信するために最適化されたネットワーク）。Google CloudSなどと組み合わせて使用することで、リクエスト元から地理的に近いサーバーにコンテンツのキャッシュを自動的に配置し、コンテンツ配信を高速化・効率化させることができる。
Cloud DNS	Google CloudのDNS。PLATEAU VIEW 4.0のPLATEAU Editorで使用しているドメインに対応するレコードは全てCloud DNSで管理されている。
Cloud Load Balancing	Google Cloudのマネージドなロードバランサ（負荷分散システム）。 PLATEAU VIEW 4.0のPLATEAU Editorで使用されるドメインのIPアドレスは全てCloud Load Balancingに向いており、リクエストのホスト（ドメイン）に応じてAPIサーバーやストレージサーバーに自動的にルーティングされる。
Cloud Logging	Google Cloudのログ収集サービス。Cloud Runなどから出力されるログを閲覧可能。システムのトラブルシューティング時に役立つ。
Cloud PubSub	Google Cloudのメッセージングサービス。アプリケーション間の連携を行うために利用される。PLATEAU CMSでは、APIサーバーとワーカーサーバー間の通信に使用している。
Cloud Tasks	Google Cloudのキューイングサービス。大量の分散タスクの実行を管理できる。 PLATEAU CMSではワーカーサーバーで行われる解凍処理のキューイング及び再試行処理制御のために使用している。
Cloud Build	Cloud Build: Google CloudのCI/CDパイプライン向けビルドプラットフォーム。 CMSでは、大規模圧縮ファイルの解凍処理の基盤として利用している。

1.1.3 PLATEAU Editor / VIEW

1.1.3.1 PLATEAU Editor

PLATEAU VIEW 4.0におけるPLATEAU Editorのシステム構成について説明する。

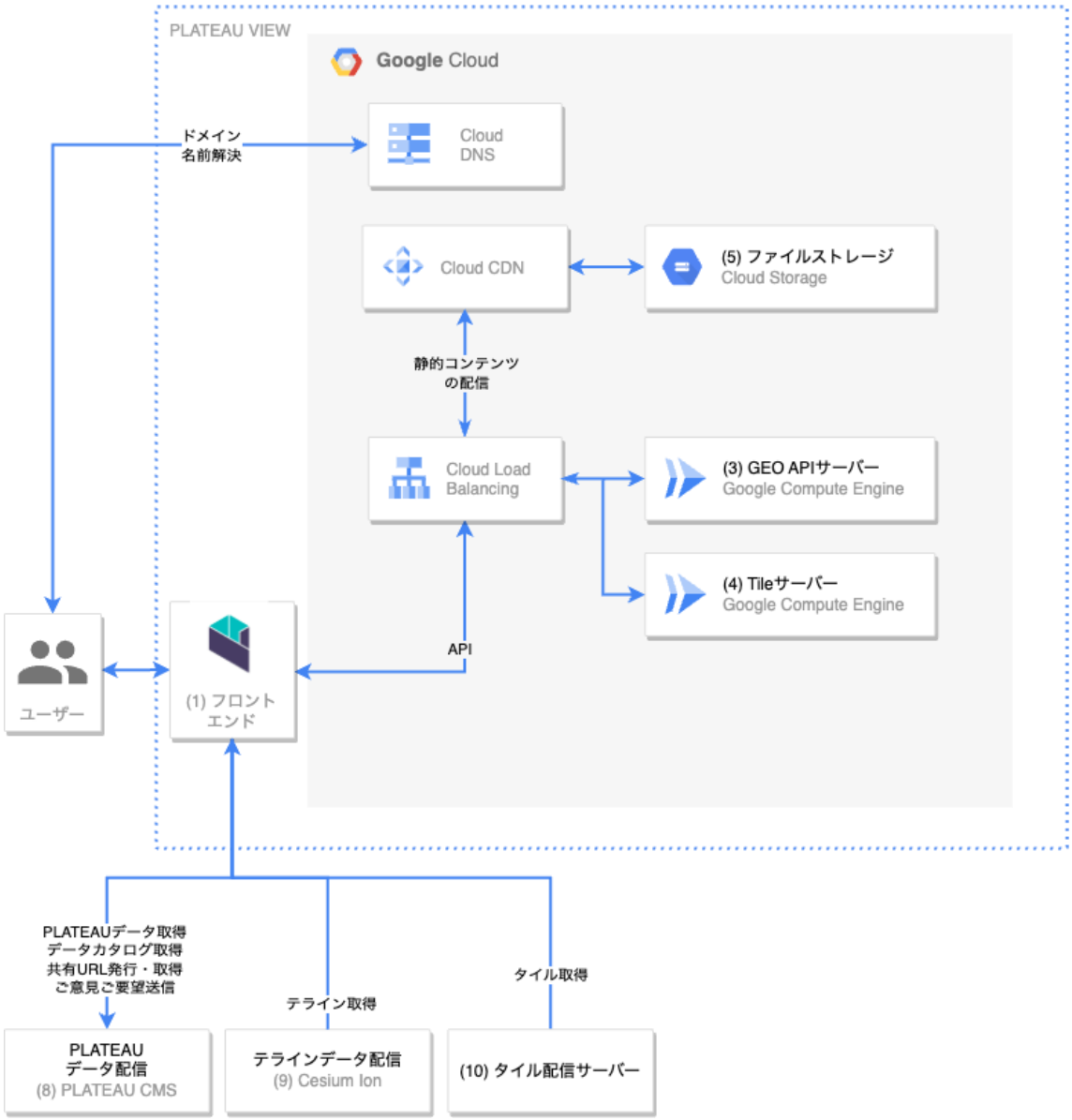
表　PLATEAU Editorのシステム構成



1.1.3.2 PLATEAU VIEW

PLATEAU VIEWのシステム構成について説明する。

表　PLATEAU VIEWのシステム構成



以下、上記構成図の各要素についてそれぞれ説明する。

なお、各コンポーネントは、保存データ量、ネットワーク転送量、CPU時間、マシンスペックなどに基づいて料金が発生する。

（1）フロントエンド

Webブラウザ上で動作する、（HTML・CSS・JavaScriptによる）フロントエンドアプリケーション。APIサーバー・Auth0・その他各種サーバーとの通信を行い、UIや地図を表示する。

（2）APIサーバー（Cloud Run）

PLATEAU EditorのAPIサーバーであり、HTTPサーバーとして外部からのリクエストを受信している。MongoDBやファイルストレージと連携して、レイヤーの保存やプロジェクトの管理・公開などの、さまざまなビジネスロジックを実行する。

APIサーバーは、Cloud Run 上で動作する。Cloud Run とは、Google Cloudで利用可能なサーバーレス（CaaS）プラットフォームであり、Dockerコンテナをデプロイすることで、サーバーの保守管理の手間なしに、アプリケーションをクラウド上で動作させることができる。同時接続リクエスト数が規定数以上に達すると自動的にコンテナが増加し、より多くのトラフィックを自動的に分散処理することができる。

Cloud Runは、デフォルト設定では、HTTPリクエストを受信して処理している間のみCPUが動作し、リクエストを処理していない時は動作を停止するため、HTTPリクエストを実際に受信し処理するために動作したCPU時間分のみが課金対象となる。この点が、常時稼働が前提となることが多いAWSのEC2やGoogle CloudのGoogle CloudEとは異なる。

（2025年3月現在）PLATEAU VIEW 4.0のPLATEAU EditorのAPIサーバーは、メモリ1GB・CPU2コアの設定で動作している。

（3）GEOサーバー（Cloud Run）

Editor及びViewerに必要な地理情報を処理するためのAPIサーバーであり、HTTPサーバーとして外部からのリクエストを受信している。住所検索や逆ジオコーディングなどEditor・Viewer上必要な機能を提供する。

（4）Tileサーバー（Google Compute Engine）

PLATEAU VIEW 4.0では、国土地理院が実験的に提供しているベクトルタイル形式の「地理院タイル」を利用している。フロントエンドで利用している、CesiumJSで利用できるよう、Tileサーバーにおいてベクトルタイルをラスター化し、Slippy Map Tilenames形式で配信している。

（5）ファイルストレージ（Google Cloud Storage）

フロントエンドのアプリケーションのソースコードや画像、ユーザーによってアップロードされたアセットファイルや、プロジェクト公開時にビルドされる情報、インストールされたプラグインのファイルを保存する、オブジェクトストレージサーバー。

Google Cloud Storage（Google CloudS）というサービスを使用しており、容量は無制限であり、自動的にスケーリングし、バックアップも自動的に行われ、データは複数拠点に分散配置される。巨大なファイルを格納・配信することが可能。

なお、PLATEAU VIEW 4.0のPLATEAU関連データセットなどのGISデータは、主にPLATEAU CMSのストレージサーバーで保存・配信されており、PLATEAU Editorで保存されているファイルの量はそれに比較してさほど大きくない。PLATEAU VIEW上で可視化されるGISデータは主にPLATEAU CMSから配信されるデータを使用している。

（６）MongoDB

MongoDBとは、ドキュメント指向のNoSQLデータベースで、データの柔軟性と拡張性が特徴。
PLATEAU Editorはデータベースとして MongoDBを使用している。

PLATEAU Editorでは、MongoDB社が提供するクラウドサービス MongoDB Atlas を利用している。保守運用が自動化されたマネージドなMongoDBが利用可能で、自動的に3台以上のサーバーから成るクラスタを構成し、データベース内のデータは自動的に各サーバーに複製され、リクエストは分散処理されるようになっている。M0からM30まで、さまざまなマシンスペックのサーバーによる可用性の高いクラスタを構築することができ、マシンスペックによって料金が変わる。

（2025年3月現在）PLATEAU VIEW 4.0のPLATEAU Editor向けには、M10クラスタを運用している。

（７）Auth0

Auth0社が提供するクラウドサービス。アカウントの管理・認証・認可を行うIDプロバイダを提供する。Auth0を使用することで、開発者はアプリケーションに安全で使いやすく信頼性の高い認証認可機能を組み込むことができる。PLATEAU EditorもAuth0を使用して認証認可機能を実現している。

（2025年3月現在）テナントに対して登録されているユーザー数に応じて課金されるが、7000ユーザーまでは無料となっている。

（８）PLATEAU CMS

PLATEAU VIEWで利用可能なデータカタログや、3D都市モデルデータをはじめとする各種GISデータ等を配信しているシステム。詳しくは2.1を参照されたい。

（９）Cesium Ion

Cesium社が提供するクラウドサービス。PLATEAU VIEW向けに独自に作成された、地球儀上の日本における地表面を表現する3Dのデータ（テラインデータ）を配信するために使用している。

（１０）タイル配信サーバー

PLATEAU VIEWの地図の設定で選択可能なベースマップについて、それぞれ以下のサーバーから配信されるタイルデータを使用している。

タイルデータとは、ユーザーからリクエストされた地図表示範囲に対して、あらかじめタイル状に分割された画像データのことであり、それらをサーバーから配信するサービスを使用することでタイルデータを取得してCesiumJS上で描画している。

- 全国最新写真 （シームレス）：PLATEAU VIEW 向けに独自に作成されたタイルデータ。
  PLATEAU VEIW 4.0向けに構築されたGoogle CloudのGoogle CloudSから配信されている。

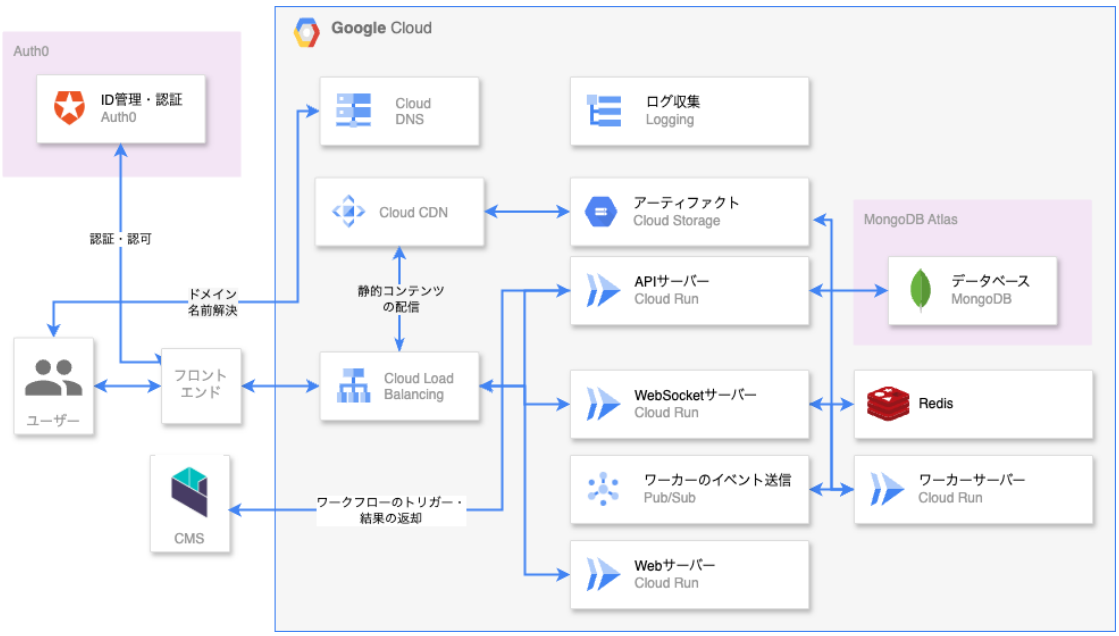
（11）その他のコンポーネント

表　その他のコンポーネント

コンポーネント名	説明
Cloud CDN	Google CloudのCDN（Content Deliver Network = ウェブコンテンツをインターネット経由で配信するために最適化されたネットワーク）。Google CloudSなどと組み合わせて使用することで、リクエスト元から地理的に近いサーバーにコンテンツのキャッシュを自動的に配置し、コンテンツ配信を高速化・効率化させることができる。
Cloud DNS	Google CloudのDNS。PLATEAU Editorで使用しているドメインに対応するレコードは全てCloud DNSで管理されている。
Cloud Load Balancing	Google Cloudのマネージドなロードバランサ（負荷分散システム）。 PLATEAU VIEW 4.0のPLATEAU Editorで使用されるドメインのIPアドレスは全てCloud Load Balancingに向いており、リクエストのホスト（ドメイン）に応じてAPIサーバーやストレージサーバーに自動的にルーティングされる。
Cloud Logging	Google Cloudのログ収集サービス。Cloud Runなどから出力されるログを閲覧可能。システムのトラブルシューティング時に役立つ。

1.1.4 PLATEAU Flow

PLATEAU Flowのシステム構成について説明する。



1.2 PLATEAU VIEW 4.0の改善点

PLATEAU VIEW 4.0では、空間IDから属性情報を取得する機能、地域メッシュからCityGMLファイルをダウンロードする機能を実装した。
空間を一意に識別する空間IDや、地域を細かく区分するメッシュコードを活用できるよう改善し、特定のエリアや地物を指定し、その属性情報やCityGMLファイルを簡単に取得できるようになった。必要なデータをピンポイントで取得できることで、検索効率が向上し、地形データの活用がよりスムーズになった。

なお、PLATEAU VIEWでの操作方法については以下を参照されたい。

https://www.mlit.go.jp/plateau/learning/tpc02-1/#p2_5

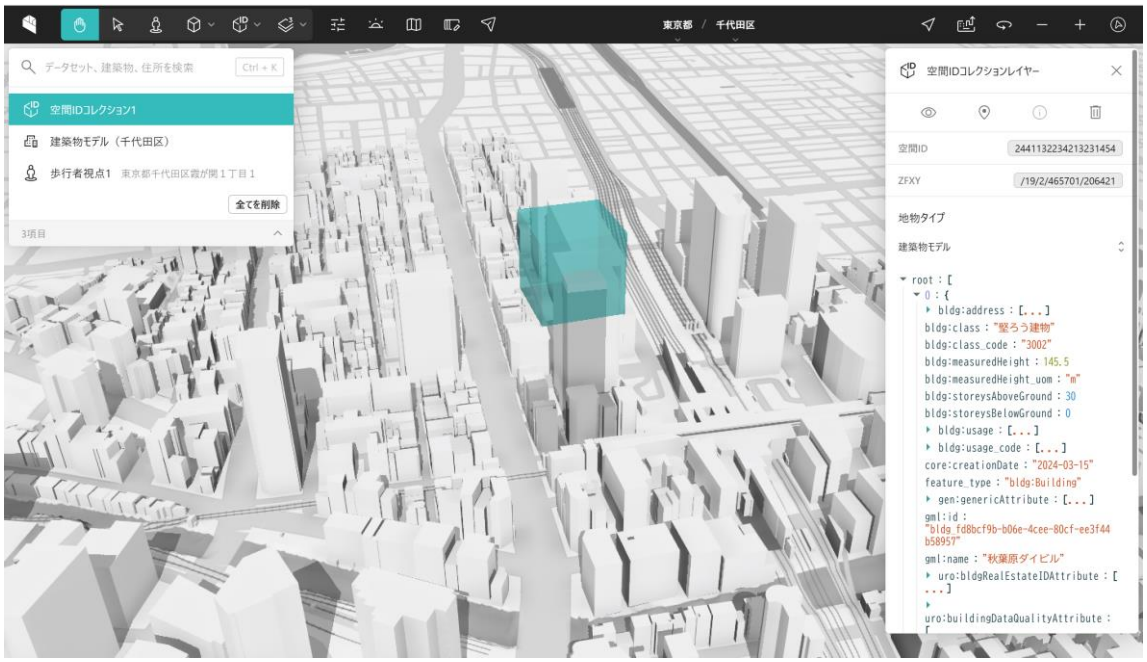
また、これらの機能はCityGML APIにより実現している。以下、CityGML APIに関して解説する。

1.2.1 空間IDから属性情報を取得する機能

PLATEAU VIEW 4.0 では、データ検索や活用の利便性を向上させるため、GUI上で空間IDを可視化し、特定エリアの属性情報やCityGMLファイルを取得できる機能を追加した。

これにより、以下の改善が実現された。

- GUI 上で空間IDを検索・表示できるようになり、3D都市モデルと空間IDの位置関係を直感的に把握できるようになった。
- 空間IDに紐づく地物（建物、道路、橋梁など）の名称、用途、高さ、建築年といった詳細な属性情報を取得できるようになった。

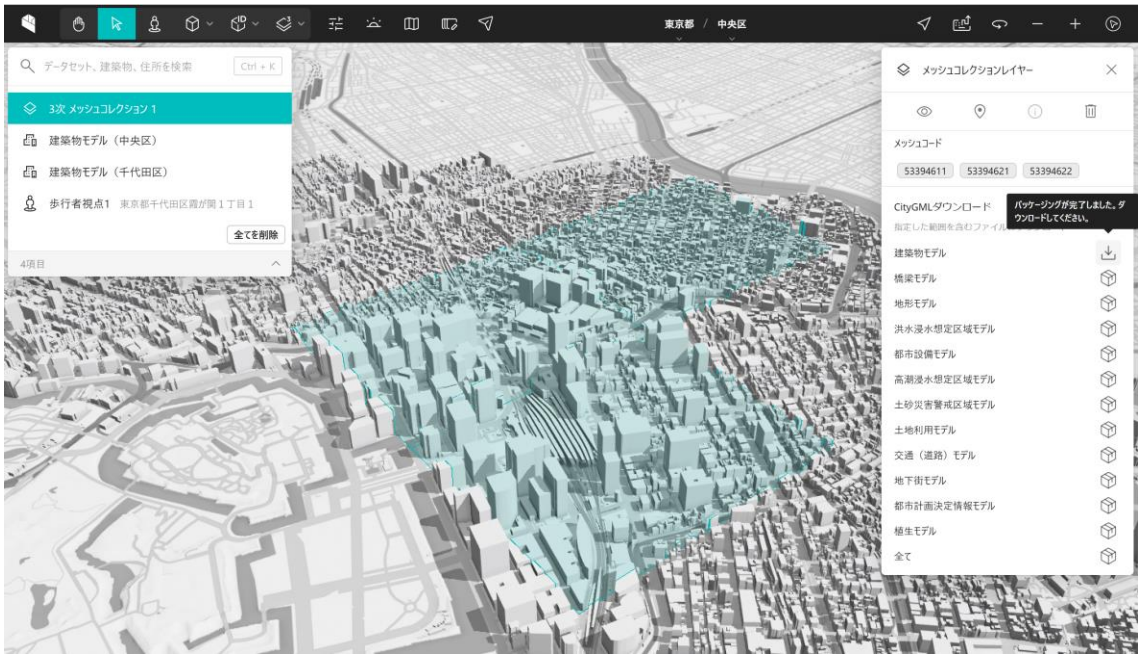


1.2.2 地域メッシュからCityGMLをダウンロードする機能

これまで、CityGMLファイルをダウンロードするためにはG空間情報センターにアクセスし、全ての地物を含む都市単位のファイルをダウンロードする必要があった。そこで、PLATEAU VIEW 4.0 では、3D都市モデルの流通性を高めるため、GUI上で地域メッシュを指定し、CityGMLファイルをダウンロードできる機能を追加した。

これにより、以下の改善が実現された。

- GUIで地域メッシュ（2次メッシュもしくは3次メッシュ単位）を選択することで特定のエリアのみをピンポイントで指定し、都市全体をダウンロードする必要がなくなった。なお、サーバーへの負荷を考慮し、一度にダウンロードできる範囲を9メッシュ分に制限している。
- 対象範囲内の各地物モデル（建築物、道路、橋梁など）を個別に選択し、それぞれの CityGMLファイルをダウンロードできるようになった。



なお、ダウンロードされるファイルには、選択したメッシュコードに対応するGMLファイルだけでなく、それに関連するCityGMLデータ一式が含まれる。具体的なファイル構成は以下の通り。

```
- ルートフォルダ　※不規則な文字列
    - 市区町村名（フォルダ）
        - udx（フォルダ）
                                - 各地物名（フォルダ）
            - メッシュコードに含まれる GML ファイル
        - schemas（フォルダ）
        - codelists（フォルダ）
        - metadata（フォルダ）
        - README.md
        - XXXXX_indexmap_op.pdf
```

これにより、CityGMLの正しい構造やスキーマとの関連性を保ったまま、データを活用することができる。

1.2.3 CityGML API

PLATEAU CMSでは、都市モデルを地物別・メッシュ単位で管理しており、個別のCityGMLファイルにURLを通じてアクセスできる環境を整備している。この特長を活かし、メッシュ単位や地物単位でCityGMLデータやその属性情報を柔軟に取得できる仕組みとしてCityGML APIを開発した。

通常、PLATEAU VIEWでは地域メッシュ単位でダウンロード対象を選択する仕様となっているが、本APIを利用することで、メッシュコードや空間ID、座標、ジオコーディング（住所検索）、自治体コードなど、さまざまな条件でデータを取得できる。
本APIは7種類のAPIを含む。なお、これらのAPIは試験的に運用されているものであり、APIスキーマなどは予告なく変更される可能性があり、SLAも保証していないことに注意されたい。

- API共通エンドポイント： https://api.plateauview.mlit.go.jp
- OpenAPI定義： https://api.plateauview.mlit.go.jp/openapi.json
- APIドキュメント： https://api.plateauview.mlit.go.jp/docs/

1.2.3.1 CityGML Files API

（1） GET /datacatalog/citygml/{conditions}

- **目的**: 自治体コードやメッシュコードなど、様々な方法で指定した範囲に含まれる各種CityGMLファイルのURLのリストを取得するためのAPI。
- **パラメータ**:
    - conditions :地理的なエリアや都市の情報をフィルタリングするための条件を指定する。詳細は下記「検索条件の種類」を参照。
- **レスポンス**:
    - **200 OK**
    - **400 Bad Request**: 無効な条件が指定された場合（例: メッシュコードや座標範囲の形式が誤っている場合）

```
{"error":"some error message"}
```

    - **404 Not Found**: 指定条件に該当するCityGMLデータが見つからない場合

```
{"error":"not found"}
```

    - **503 Service Unavailable**: ジオコーディングサービスにアクセスできない場合

（2）検索条件の種類

このAPIでは、 :conditions で地理的なエリアや都市の情報をフィルタリングするための条件を指定する。
使用例: m:533945, s:spatialID, r:135.5,35.0, g:locationName
:conditionsパラメータは以下の条件タイプに対応している。

1. **メッシュコード (m)**
    - **概要**: カンマ区切りで複数のメッシュコードに基づいてエリアを指定。2次メッシュ（6桁）・3次メッシュ（8桁）・1/2メッシュ（9桁）・1/4メッシュ（10桁）に対応する。メッシュコードはハイフンで区切らず数値だけで表すこと。
    - **形式**: m:<メッシュコード>
    - **例**: m:533944,533945
    https://api.plateauview.mlit.go.jp/datacatalog/citygml/m:533935
2. **メッシュコード厳密検索 (mm)**
    - **概要**: 1とほぼ同じだが、1と比べ、メッシュコードの桁数が完全に一致するCityGMLのみを取得する。2次メッシュの場合は2次メッシュのデータセットのみが、3次メッシュの場合は3次メッシュのCityGMLのみがヒットする。
    - **形式**: mm:<メッシュコード>
    - **例**: mm:533944,533945
    https://api.plateauview.mlit.go.jp/datacatalog/citygml/mm:533935
3. **空間ID (s)**
    - **概要**: カンマ区切りで複数の空間IDを使用してエリアを指定。
    - **形式**: s:<空間ID>,<空間ID>,...
        - 空間IDは、 /z/x/y 、 /z/f/x/y 、およびタイルハッシュの表記に対応。
    - **例**:
    https://api.plateauview.mlit.go.jp/datacatalog/citygml/s:/18/1/232853/103220（ /z/f/x/y ）
4. **座標範囲 (r)**
    - **概要**: 座標範囲を指定してエリアを指定。
    - **形式**:
        - 2要素（lng, lat）で中心点を指定する場合: r:lng,lat
        - 4要素（lng1, lat1, lng2, lat2）で範囲を指定する場合: r:lng1,lat1,lng2,lat2
        注意:lng1 < lng2, lat1 < lat2 となるように指定する必要がある。
    - **例**:
        - 中心点の指定: r:139.7375,35.65833333333333
        https://api.plateauview.mlit.go.jp/datacatalog/citygml/r:139.7385,35.65933333333333
        - 範囲の指定: r:139.7375,35.65833333333333,139.74,35.66
        https://api.plateauview.mlit.go.jp/datacatalog/citygml/r:139.7375,35.65833333333333,139.74,35.66
5. **ジオコーディング (g)**
    - **概要**: ジオコーディングAPIを用いてエリアを指定する。
    - **形式**: g:<ロケーション名>
    - **例**: g:千代田区
    https://api.plateauview.mlit.go.jp/datacatalog/citygml/g:千代田区
6. **自治体コード**
    - **概要**: カンマ区切りで複数の自治体コードを指定する。
    - **形式**: cityId1,cityId2,...
    - **例**: 13999
    https://api.plateauview.mlit.go.jp/datacatalog/citygml/13999

1.2.3.2 CityGML Pack API

CityGMLのURLのリストを入力すると、関連するCityGMLファイルを含むzipファイルを生成（パック）するAPI。
サーバーの負荷を抑制するためタイムアウトなどの制限が設けられており、また一度作成されたzipファイルは一定時間経過後、削除されることがあることに注意。

（1）POST /citygml/pack

- **目的**: CityGMLファイルを含んだzipファイルの非同期作成をリクエストする。
- **リクエストボディ（JSON）**:
    - urls :ファイルのURLのリストを指定。PLATEAU CMSから配信されるファイル（assets.cms.plateau.reearth.io）以外のURLはエラーになる。
- **レスポンス**:
    - **200 OK**: リクエストが受理され、ステータス確認のためのIDが返却される。この id フィールドの値は、後で GET /pack/{id}/status を使ってパッキングの状態を確認する際に使用する（パックID）。

```
{ “id”:
"2df41c35a57f7e63a04f422a6843faf047dac428c386bfbc7e2aad393a76472c" }
```

    - **400 Bad Request**: 無効なリクエストやURLエラー。
        - リクエストボディが無効な場合

```
{
  "error": "invalid request body",
  "reason": "invalid JSON"
}
```

        - 無効なURLが含まれている場合

```
{
  "error": "invalid url",
  "url": "<http://example.com/invalid_file.gml>"
}
```

        - ドメインが無効な場合

```
{
  "error": "invalid domain",
  "url": "http://another-domain.com/file.gml"
}
```

    - **500 Internal Server Error**: サーバーエラー。
        - サーバーエラーでパックリクエストが処理できなかった場合

```
{ "error": "failed to enqueue pack job" }
```

- 呼び出し例

```
curl https://api.plateauview.mlit.go.jp/citygml/pack ¥
  --json '{
    "urls": [
      "https://assets.cms.plateau.reearth.io/assets/59/984b81-8516-4e13-a8f5-
2ca24e272c3e/13101_chiyoda-
ku_city_2023_citygml_1_op/udx/bldg/53394509_bldg_6697_op.gml",
      "https://assets.cms.plateau.reearth.io/assets/59/984b81-8516-4e13-a8f5-
2ca24e272c3e/13101_chiyoda-
ku_city_2023_citygml_1_op/udx/bldg/53394518_bldg_6697_op.gml"
    ]
  }'
```

（2）GET /citygml/pack/{id}/status

- **目的**: 指定したIDのCityGMLパックプロセスの状態を確認する。
- **パラメータ**:
  - id:パックID
- **レスポンス**:
  - **200 OK**: 状態を返却（accepted, processing, succeeded, failed）。

```
{
  "status": "succeeded",
  "startedAt": "2025-02-19T05:10:20.907Z",
  "progress": 0.1
}
```

    - status:パック処理のステータス
      - accepted - リクエストが受理された。
      - processing - パック処理が進行中。
      - succeeded - パック処理が完了。
      - failed - パック処理が失敗。タイムアウトの場合も含む。
    - startedAt : パック処理の開始日時
    - progress : パック処理の進捗率。0～1の範囲で示す。完了時は1。
  - **404 Not Found**: ファイルが存在しない場合のエラー。指定したIDのファイルが存在しないか、パック処理が開始されていない場合に返される。

```
{ "error": "not found" }
```

- 呼び出し例

```
curl
https://api.plateauview.mlit.go.jp/citygml/pack/2df41c35a57f7e63a04f422a6843faf047dac428
c386bfbc7e2aad393a76472c/status
```

（3）GET /citygml/pack/{id}.zip

- **目的**: 指定したIDのCityGMLパックファイルをダウンロードする。
- **パラメータ**:
  - id:パックID
- **レスポンス**:
  - **302 Found**: リダイレクトしてファイルのダウンロードが開始。
  - **404 Not Found**: ファイルが存在しない場合のエラー。

```
{ "error": "not found" }
```

  - **400 Bad Request**: ファイルの状態が不正の場合、またはパック処理が完了していない場合。
- 呼び出し例:

```
curl -L -X GET
https://api.plateauview.mlit.go.jp/citygml/pack/2df41c35a57f7e63a04f422a6843faf047dac428c386bfbc7e2aad393a76472c.zip -o citygml_pack.zip
```

1.2.3.3 CityGML 属性API

（1）GET /citygml/attributes

- **目的**: CityGMLのURLとgml_idのリストを入力すると、属性情報をJSONで返却するAPI。
- **クエリパラメータ**:
  - url:CityGMLファイルのURLを指定。1つのみ。PLATEAU CMSから配信されるファイル（assets.cms.plateau.reearth.io）以外のURLはエラーになる。
  - id:gml:id をカンマ区切りで複数指定。
  - skip_code_list_fetch:何らかの値が指定されていれば、コードリスト取得をスキップし生のコードを返す。
- **レスポンス**:
  - **200 OK**: リクエストが受理され、属性情報がJSON形式で返却される。レスポンス内容は以下の呼び出し例を参照。なお、3D Tilesの属性情報に入っているmeshcodeはレスポンス内容に含まない。
  - **400 Bad Request**: 無効なリクエストやURLエラー。
    - 無効なURLが含まれている場合

```
{
  "error": "cannot fetch",
  "url": "http://example.com/invalid_file.gml"
}
```

    - CityGMLを取得できなかった場合

```
{
  "error": "cannot fetch",
  "url": "http://example.com/invalid_file.gml"
}
```

    - ドメインが無効な場合

```
{
  "error": "invalid domain",
  "url": "http://another-domain.com/file.gml"
}
```

- **500 Internal Server Error**: サーバーエラー。
    - XMLのパースに失敗した場合。

```
{
  "error": "internal",
  "url": "http://example.com/invalid_file.gml"
}
```

- 呼び出し例
https://api.plateauview.mlit.go.jp/citygml/attributes?url=https://assets.cms.plateau.reearth.io/assets/b0/cf0c83-7df3-40c4-a95d-7cc66a801cc6/22100_shizuoka-shi_city_2023_citygml_1_op/udx/bldg/52382287_bldg_6697_psc_op.gml&id=bldg_c77c3e2b-ffdc-4b1a-91bf-185a1b46a4d1,bldg_2eb12f7a-c5d9-4145-9609-a6a0f5824368

1.2.3.4 CityGML 空間ID検索API

（1）GET /citygml/features

- **目的:** CityGMLのURLと空間IDのリストを入力すると、その空間ID内に含まれる地物のgml_idのリストを返す。
- **クエリパラメータ**:
    - sid:空間IDでカンマ区切りで指定可能。/z/x/y・/z/f/x/y・ハッシュタイルが指定可能。
- **レスポンス**:
    - **200 OK**: CityObjectMemberのgml:idが複数返却される。

```
{
  "featureIds": [
    "bldg_09f2c415-e668-4de4-880a-40cfd91a57b1",
    "bldg_1d1c7c3d-65b0-4228-a4a7-041a5a960bc4",
    "bldg_55958432-0b29-484b-a4f5-7b366e94f77a",
    "bldg_a917f23b-8c8e-4491-b184-4ceeab3e8f77"
  ]
}
```

- **呼び出し例**
https://api.plateauview.mlit.go.jp/citygml/features?url=https://assets.cms.plateau.reearth.io/assets/59/984b81-8516-4e13-a8f5-2ca24e272c3e/13101_chiyoda-ku_city_2023_citygml_1_op/udx/bldg/53394600_bldg_6697_op.gml&sid=/18/0/232840/103234,/18/0/232840/103235

1.2.3.5 CityGML 空間ID属性API
（1）GET /citygml/spatialid_attributes

- **目的**: 空間IDと地物型のリストを入力すると、その範囲に含まれる地物の属性情報をJSON形式で返却するAPI。1・4・3のAPIを順番に呼び出す手間を省くことができる。
- **クエリパラメータ**:
    - sid :空間IDをカンマ区切りで複数指定可能。最低1つ以上必要。/z/x/y・/z/f/x/y・ハッシュタイルが指定可能。
    - type :地物型をカンマ区切りで複数指定可能。最低1つ以上必要。
- **レスポンス**:
    - **200 OK**: リクエストが受理され、属性情報がJSON形式で返却される。レスポンスのスキーマはCityGML属性APIと同じ。
    - **400 Bad Request**: 無効なリクエストやURLエラー。
        - sidパラメータが不足の場合（最低1つは必要）

```
{ "error": "sid parameter is required" }
```

        - typeパラメータ不足の場合（最低1つは必要）

```
{ "error": "type parameter is required" }
```

        - CityGMLを取得できなかった場合

```
{
  "error": "cannot fetch",
  "url": "http://example.com/invalid_file.gml"
}
```

    - **404 Not Found**: 検索したが見つからなかった場合。
        - CityGMLが見つからなかった場合

```
{ "error": "not found" }
```

        - CityGMLに指定された地物型のち物が存在しなかった場合

```
{ "error": "no citygml files for the given types" }
```

        - 指定した空間ID内に地物が見つからなかった場合

```
{ "error": "no features found" }
```

    - **500 Internal Server Error**: サーバーエラー。
        - XMLのパースに失敗した場合

```
{
  "error": "internal",
  "url": "http://example.com/invalid_file.gml"
}
```

        - CityGMLから属性の抽出に失敗した場合

```
{ "error": "failed to extract attributes
```

        - その他内部エラー

```
{ "error": "internal" }
```

- 呼び出し例

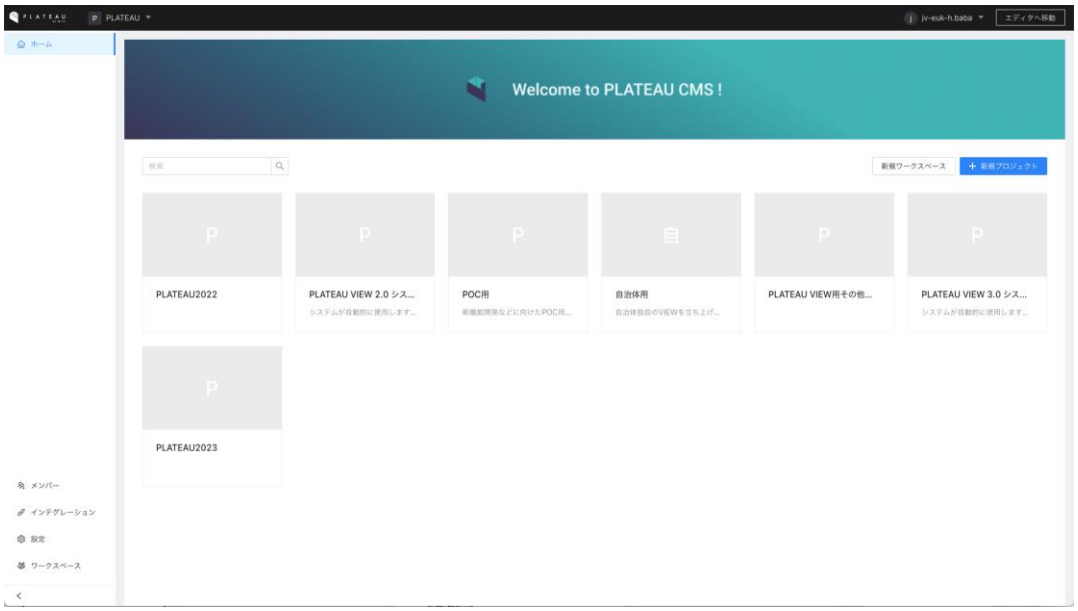
https://api.plateauview.mlit.go.jp/citygml/spatialid_attributes?sid=/18/0/232840/103234,/18/0/232840/103235&type=bldg,veg,tran

第２章　PLATEAU VIEW 4.0の機能

2.1 PLATEAU CMS

2.1.1 PLATEAU CMSとは

1.1のシステム構成で述べたように、PLATEAU VIEW 4.0のシステムの機能は、大きく「データの管理」と「データの可視化」に分かれる。PLATEAU CMSは、このうち「データの管理」を担うシステムのことを指す。



CMSとは「コンテンツ管理システム（Content Management System）」の略で、一般的な概要は以下のとおりである。

> コンテンツ管理システム（CMS）は、企業がデジタルコンテンツを管理するのに役立ちます。チーム全体がこれらのシステムを使って、コンテンツの作成、編集、整理、公開を行うことができます。コンテンツを保存する単一の場所として機能し、組み込み（または設計された）ワークフローを使用して、共同デジタルコンテンツ管理および作成のための自動化されたプロセスを提供します。役割に応じて、個人にはさまざまな特権と責任が与えられます。例えば、著者は作品を投稿し保存することができますが、編集者は作品を修正し公開することができます。管理者は、こうした作業をすべて行えるだけでなく、組織内の他の人にコンテンツの更新や改訂の許可を与えることもできます。
>
> *CMS*では、最小限の技術コストで*Web*サイトや*Web*サイトのコンテンツを作成・管理できるため、プロジェクト・マネージャーやトラフィック・マネージャーのような役割を果たす必要なしに、より優れたコンテンツの作成に集中することができます。*CMS*は、コンテンツ管理のための簡単で費用対効果の高いソリューションを提供します。これにより、企業は専任のコンテンツ開発チームに投資しなくても、コンテンツを管理・配信することができます。
>
> 引用元：https://www.oracle.com/jp/content-management/what-is-cms/

この中でもPLATEAU CMSは、管理対象のコンテンツをAPIを通じて提供することを前提とした「ヘッドレスCMS」に位置付けられる。管理者だけでなく、地方自治体・受託事業者やユースケースデータの登録者などさまざまな事業者がPLATEAU VIEW 4.0で公開するPLATEAU関連データセットを一元管理し、APIとして公開することができるシステムである。

（1）使用ソフトウェア・サービス

上記システムを構築するために、オープンソースソフトウェア（OSS）と、有償のクラウドサービスを組み合わせて利用している。クラウドサービスの詳細については次項の各コンポーネントの説明で述べる。

表　使用ソフトウェア・サービス一覧

項目	項目	説明
FME	有償クラウドサービス	データの品質検査と変換を行うアプリケーション。詳しい説明は3.3を参照。
G空間情報センター	オープンデータ・ストレージサービス	官民問わずさまざまな主体により整備・提供される多様な地理空間情報を集約し、利用者がワンストップで検索・ダウンロードし利用できる、産学官の地理空間情報を扱うプラットフォーム。OSSのCKANを用いて構築されている。
Auth0	有償クラウドサービス	Auth0社が提供するクラウドサービス。アカウントの管理・認証・認可を行うIDプロバイダを提供する。Auth0を使用することで、開発者はアプリケーションに安全で使いやすく信頼性の高い認証・認可機能を組み込むことができる。PLATEAU Editorも Auth0を使用して認証・認可機能を実現している。
SendGrid	有償クラウドサービス	Twillio社が提供するクラウドサービス。企業や開発者がアプリケーションやWebサイトから大量のメールを配信するために使用される。PLATEAU CMSではご意見ご要望のメール送信で使用している。
Google Cloud	有償クラウドサービス	Google社が提供するクラウドコンピューティングプラットフォーム。PLATEAU CMSを動作させるためのサーバーや、ファイルを保存するためのサーバーをGoogle Cloud上に構築している。
MongoDB Atlas	有償クラウドサービス	MongoDB社が提供するクラウドサービス。保守運用が自動化されたマネージドなMongoDBを提供している。MongoDBとは、ドキュメント指向のNoSQLデータベースで、データの柔軟性と拡張性が特徴。PLATEAU CMSはデータベースとして MongoDBを使用している。

（2）使用技術・ライブラリ

PLATEAU CMSはその他さまざまな技術を組み合わせて構築されている。

PLATEAU CMSで内部的に利用されている技術・ライブラリ等

項目	説明
Go	プログラミング言語の1つで、Google社によって開発された。構文がシンプルであり、かつ処理が高速な言語であり、主にバックエンド開発に用いられる。PLATEAU CMSのバックエンド実装に利用されている。
TypeScript	プログラミング言語の1つで、Microsoftによって開発された。JavaScriptに静的型付けを加えたスーパーセットであり、大規模システムの開発に用いられる。PLATEAU CMSではフロントエンド向け開発に利用されている。
React	Meta社によって開発されたUI構築のためのJavaScriptライブラリ。特に大規模かつ複雑なUI実装において利用される。PLATEAU CMSではフロントエンド向け開発に利用されている。
CesiumJS	デジタル3D地球儀上にさまざまな情報を描画することができる地図エンジン。Webブラウザ上で動作し、WebGLを用いて描画を行うため、PCやスマートフォンで閲覧することができる。PLATEAU CMS上では、データプレビューで使用している。
Resium	React上でCesiumJSを手軽に利用可能にするコンポーネントを提供するライブラリ。Eukarya開発。PLATEAU CMS上では、データプレビューで使用している。
GraphQL	API向けに作られたクエリ言語及びランタイムを指す。WebAPIの開発において、RESTなどの方式と比較して、より柔軟かつ効率的なAPIの提供を可能にする。PLATEAU CMSではバックエンドとフロントエンド間の通信に利用されている。
Terraform	HashiCorp社によって開発された、infrastructure-as-codeを実現するためのソフトウェアであり、サーバー構成をコードとして宣言的に管理をできるようにする。

（3）PLATEAU CMSの主な機能

PLATEAU CMSでは主に以下の機能が利用可能である。詳しい使い方は、2.1.2を参照されたい。

- ワークスペースの作成・ユーザーの招待
- プロジェクトの作成
- スキーマの定義
- コンテンツの登録・編集
- アセットの登録・Zipファイルの解凍・プレビュー
- コンテンツの公開リクエスト
- コンテンツの公開（公開APIとしてコンテンツを公開可能）
- インテグレーションの作成・インストール（外部システムとの連携が可能で、本システムではFME Flowとの連携等で利用。）

（4）対応データフォーマット等

アセットの対応データフォーマット

PLATEAU CMSでは、全てのフォーマットの静的ファイルをアップロードが可能であるが、特に以下のファイルフォーマットに関してはプレビュー機能をサポートしている。

- 画像データ
    - PNG
    - JPEG
    - SVG
    - GIF
- GISデータ
    - GeoJSON
    - CZML
    - KML
    - Mapbox Vector Tiles（MVT）
    - glTF（glb）
    - 3D Tiles

スキーマのフィールド型

スキーマのフィールド型としては、以下のデータ型をサポートしている。

- テキスト: 短文向けのフィールド
- テキストエリア: 長文向けのフィールド
- マークダウン: マークダウンのフィールド
- アセット: アセットをリンクするためのフィールド
- 日付: 日付のフィールド
- 真偽値: 真偽値のフィールド
- 選択: 選択式のフィールド
- 整数値: 数値のフィールド
- URL: URLのフィールド
- 参照: 他のモデルを参照するフィールド
- グループ: 複数のフィールドをまとめるフィールド

PLATEAU VIEW 4.0対応データフォーマット一覧（1/2）

フォーマット				説明
	単体	Zip 7Z	URL 指定	
GeoJSON	✓		✓	•JSON形式で記述されるGISファイルフォーマット。点、線、面のベクトルデータの表示に対応している。 ＜対応可能事項＞ •RFC7946で定義されたGeoJSON •ジオメトリのCRS：WGS84（EPSG:4326） ＜特記事項＞ 以下の形式については未対応である。 •GeoJSON 2008 •TopoJSON •CRSの指定*1 •標準仕様外のジオメトリ（Circleなど） *1：座標参照系（CRS:Coordinate Reference System）
CZML	✓		✓	•CesiumJS上でのデータ表現に対応したJSON形式のGISファイルフォーマット。 •CesiumJSの機能を使用してCZMLを表示するため、CZMLの仕様内でPLATEAU VIEW 4.0固有の制約はない。 ＜特記事項＞ •CZMLとその他関連するファイルをZipまたは7zファイルに圧縮・同梱することで、CZMLから相対パスで参照可能な別のデータセットを同梱することができる。この場合、CZMLはzipファイルのルート直下に置かれており、Zipファイル名と拡張子を除く部分が同じである必要がある。 （詳細は”CZMLをZip化する際のディレクトリ構成サンプル”を参照） •CZMLのdescription中では、相対パスや相対URLは使用できない。インフォボックス内に画像を表示させたい場合は、インターネット上で公開されている画像を絶対URL（httpまたはhttpsから始まるURL）で指定するか、CMSに画像だけを先にアップロードしてその画像の絶対URLを取得することで、使用することができる。
3D Tiles	✓		✓	•複数ファイルから構成されるデータフォーマットであるため、CMSにアップロードする場合は、それらをZipまたは7zファイルに圧縮する必要がある。 （詳細は”3DTilesをZip化する際のディレクトリ構成サンプル”を参照） ＜特記事項＞ •圧縮する際には、ルート直下にtileset.jsonファイルを格納する必要がある。 ＜例外＞ •b3dmなど他のファイルはtileset.json内で相対パスが正しく定義されていれば、フォルダを挟んでも問題ない。
MVT		✓	✓	•拡張子は.mvt に対応する。 •複数ファイルから構成されるデータフォーマットなので、CMSにアップロードする場合は、それらをZipまたは7zファイルに圧縮してから必要がある。その場合、ルートから {z}/{x}/{y}.mvt のようにファイルを配置する。 ＜特記事項＞ •CMSでレイヤー名を指定しないと表示されない。レイヤー名はカンマ区切りで複数指定可能である。 •FMEでMVTへのデータ変換を行った際に出力されるmetadata.jsonに対応している。 •metadata.jsonを同梱する場合には、VIEWでのカメラボタンクリック時にカメラ位置をその内容に応じて自動的に移動することができる。 •metadata.jsonがルートに存在しない場合、カメラ移動は自動的に行われないため、予めEditorでカメラ位置を手動設定する必要がある。 •URL指定の場合、 {z}/{y}/{x}.mvt の指定がURL中にない場合は、それがURLの最後に自動的に付加されたものと同じ扱いになる。

PLATEAU VIEW 4.0対応データフォーマット一覧（2/2）

フォーマット				説明
	単体	Zip 7Z	URL 指定	
Tiles		✓	✓	•複数ファイルから構成されるデータフォーマットであり、XYZ軸で分割された画像タイルである。CMSにZip形式でアップロードする場合は、ルートから{z}/{x}/{y}.png のようにファイルを配置する。 <特記事項> •URL指定の場合、 {z}/{y}/{x}.png の指定がURL中にない場合は、それらの文字列がURLの最後に自動的に付加される。
WMS (Web Map Service)			✓	<特記事項> •URLで指定する場合、レイヤー名の指定が必須となる。レイヤー名はカンマ区切りで複数指定可能。
TMS (Tile Map Service)		✓	✓	•複数ファイルから構成されるデータフォーマットであり、CMSにZip形式でアップロードする場合は、ルートから {z}/{x}/{y}.png のようにファイルを配置する。 <特記事項> •URLで指定する場合は、tilemapresource.xmlへのURLではなく、その親を指定する。 【誤】https://example.com/tms/tilemapresource.xml 【正】 https://example.com/tms •PNG画像のみ対応。拡張子は.png。tilemapresource.xmlが必須となる。
glTF（glb）	✓		✓	•拡張子は.gltf と.glb に対応。 <特記事項> •モデルの座標をCesium内部の座標系に合わせておく必要がある。事前に位置合わせを済ませたデータを使用すること。 •Web上で公開されているような通常のglTFでは座標系が異なるため正しく表示されない。
CSV	✓		✓ 単体のみ	<特記事項> •CSVファイル内のデータの1行目はヘッダとして扱われるため、各カラムの名前指定は必須となる。 •ジオメトリはポイントのみ対応する。ポイントの座標は、緯度・経度・高さでカラムを分けて数値で指定する。CMS上での登録時、これらのカラム名は自由だが、VIEWで正しく表示するにはEditorでのコンポーネント設定が必要であり、Editorでどのカラムを緯度・経度・高さとして扱うかをそれぞれ指定する。高さカラムは省略可能で、省略時は0として扱われる。 Editor側のカラム名は以下の通りである。 •ジオメトリ（緯度、経度、高さとして読み込むカラム）：at, lng, lon, height, alt, •スタイル（地図上のポイントのサイズと色）：pointSize, pointColor

3D Tilesの属性の仕様検討

2.1.2にあるように、CMSとFME Flowが連携することで、CityGML形式の3D都市モデルデータを3D Tiles等のデートフォーマットへ変換を行っている。ここでは、変換後3D Tilesデータの属性について検討した内容を記載する。

3D Tilesへ適用するスタイル

CesiumJS上での3D Tilesデータへの色分け・絞り込みなどのスタイル適用は、通常3D Tiles Styling languageを用いて行われることが一般的である。3D Tiles及び3D Tiles Styling languageの詳細はOGoogle Cloudの仕様書を確認すること。この3D Tiles Styling languageでは、3D Tiles内のデータが持つ属性値を参照してスタイルを適用することができるが、一階層目のJSONプロパティを利用したスタイル適用しかできない。例えば、以下のような属性を持つ地物があった場合には、「heightが100ならば黒色にする」というスタイルは設定できるが、「”others”の中にある”用途”が”業務施設”の場合は黒色にする」というスタイルは設定ができない。

```
{
 “height”: 100,
 “others”: {
  “用途”: “業務施設”
 }
}
```

tileset.jsonに記載される属性

tileset.jsonは対象の3D Tilesデータに関するメタデータなどを保持しており、これを利用することで実際の地物データを読み込むことなくどういった属性が存在するかをアプリケーションが知ることができる。しかし、tileset.jsonとして格納される属性も前述した一階層目のJSONプロパティのみであるため、アプリケーション側で制御したい属性は一階層目のJSONプロパティとして保持し、tileset.jsonにも記載することが望ましい。

PLATEAU VIEWでのスタイル

PLATEAU VIEWでは、3D Tiles Styling languageのこうした制約を排除し、より柔軟なスタイルができるよう独自のスタイル適用システムを実装している。具体的には、スタイルに利用する属性をJSONPathで指定できるようにしており、上述の例において、 「”others”の中にある”用途”が”業務施設”の場合は黒色にする」というスタイルも設定が可能である。

変換後3D Tilesデータの属性を検討する観点

これらを踏まえて3D Tilesデータの属性を検討する際には以下の観点で検討を行なった。

- データを利用する開発者にとっての利便性：3D都市モデルを利用するほとんどの利用者は、3D Tilesへのスタイル適用をCesiumJS標準の3D Tiles Styling languageで行うと予想される。よって、スタイルに特に利用されうる属性は一次元の属性として保持した方が利便性は向上する。一方で、3D都市モデルデータは大量の属性情報を持つため、すべての属性を一次元の属性として保持すると冗長すぎて利便性が下がってしまう。
- 3D都市モデルとしての属性の網羅性担保：3D都市モデルは様々な原典データをもとに豊富な属性情報を保持しているため、利便性を理由に属性が減らした3D Tilesデータを整備すると、豊富な属性が利用できなくなってしまう。

3D Tilesの属性

これらを踏まえ、PLATEAU VIEWでは以下の方針で3D Tilesの属性を整備することにした。

- 3D都市モデルに含まれるobjectlist（整備対象の属性一覧を含むExcelファイル）で定義されている属性で、データ作成上必須もしくは原則整備とされている項目を一次元な属性として展開する。
- 属性の網羅性担保のために、その他全ての属性を「attributes」という項目にJSON形式で格納する。

以下は3D Tilesとして変換後の建築物属性の例である。


```
{
 "bldg:measuredHeight": 166.8,
 "bldg:storeysAboveGround": 35,
 "bldg:storeysBelowGround": 3,
 "uro:BuildingDetailAttribute_uro:buildingRoofEdgeArea": 5044.6561,
 "uro:BuildingDetailAttribute_uro:surveyYear": 2021,
 "_lod": 1,
 "_x": 139.76965653701913,
 "_y": 35.677316549080686,
 "_xmin": 139.76899546137153,
 "_xmax": 139.77031761266673,
 "_ymin": 35.676873166608715,
 "_ymax": 35.67775993155265,
 "_zmin": 3.89,
 "_zmax": 162.54,
 "meshcode": "53394611",
 "feature_type": "bldg:Building",
 "city_code": "13102",
 "city_name": "東京都中央区",
 "gml_id": "bldg_fe0ea6d6-70d5-4b78-b676-6f01387a98ff",
 "attributes": {
  "meshcode": 53394611,
  "feature_type": "bldg:Building",
  "gml:id": "bldg_fe0ea6d6-70d5-4b78-b676-6f01387a98ff",
  "core:creationDate": "2024-03-15",
  "gen:genericAttribute": [
   {
    "type": "string",
    "name": "延べ面積換算係数",
    "value": "1.00"
   }
 ]
}
```

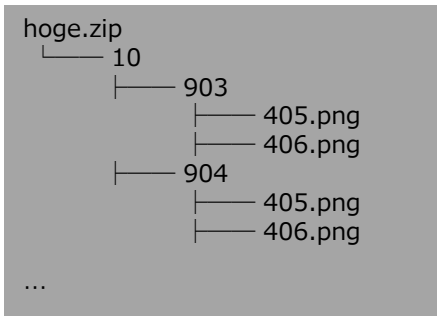
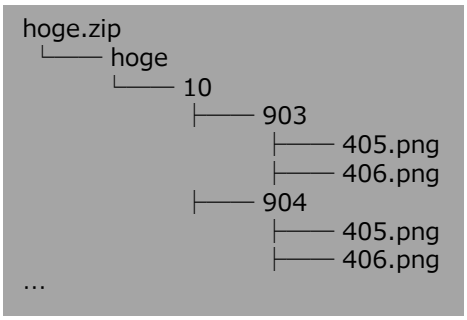
① CZMLをZip化する際のディレクトリ構造サンプル

良い例	悪い例
hoge.zip ├── hoge.czml ├── icon1.png └── icon2.png 例１）CZMLがルート直下に存在しており、ファイル名がZip/7zと同じである。	hoge.zip └── hoge 　　├── hoge.czml 　　├── icon1.png 　　└── icon2.png 例１）CZMLがルートに存在せずフォルダーを1つ挟んでいる。 hoge.zip ├── foobar.czml ├── icon1.png └── icon2.png 例２）CZMLの名前がZip/7zの名前と異なる。

② 3D TilesをZip化する際のディレクトリ構造サンプル

良い例	悪い例
hoge.zip ├── tileset.json ├── 0.b3dm └── 1.b3dm 例１）tileset.jsonがルートに存在する。（b3dmなど他のファイルはtileset.json内で相対パスが正しく定義されていれば、フォルダーを挟んでも問題ない）	hoge.zip └── hoge 　　├── tileset.json 　　　　├── 0.b3dm 　　　　└── 1.b3dm 例１） tileset.jsonがルートに存在せずフォルダーを1つ挟んでいる。

③　MVTをZip化する際のディレクトリ構造サンプル

良い例	悪い例
 例１）ズームレベル以下のフォルダーがルート直下に存在している。	 例１）ズームレベル以下のフォルダーがルートに存在せずフォルダーを1つ挟んでいる。

（5）PLATEAU CMSへアップロードするデータの命名規則

CMSへアップロードするアセットには命名規則がある。これは、CMSとFMEでの品質検査及びデータ変換において、ファイル名が重要な役割を果たすからである。CMSへアップロードするファイルは以下の命名規則に従う必要がある。

CMSへアップロードするデータの命名規則（1/2）

データ分類	データ	命名規則	記載例	CMSへのアップロード主体
都市モデルデータ	コードリスト	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_codelists.Zip	22211_iwata-shi_city_2023_citygml_1_op_codelists.Zip	ユーザー
	スキーマ	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_schemas.Zip	22211_iwata-shi_city_2023_citygml_1_op_schemas.Zip	ユーザー
	仕様	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_specification.Zip	22211_iwata-shi_city_2023_citygml_1_op_specification.Zip	ユーザー
	メタデータ	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_metadata.Zip	22211_iwata-shi_city_2023_citygml_1_op_metadata.Zip	ユーザー
	3D 都市モデル整備範囲図	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_indexmap.pdf	22211_iwata-shi_city_2023_citygml_1_op_indexmap.pdf	ユーザー
	地物型データ（建築物モデル）	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_bldg.Zip	22211_iwata-shi_city_2023_citygml_1_op_bldg.Zip	ユーザー
	地物型データ（土地利用モデル）	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_luse.Zip	22211_iwata-shi_city_2023_citygml_1_op_luse.Zip	ユーザー
	地物型データ（建築物モデル）（政令指定都市以外）	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_bldg_3dtiles_[lod].Zip	22211_iwata-shi_city_2023_citygml_1_op_bldg_3dtiles_lod1.Zip	FME
	地物型データ（建築物モデル）（政令指定都市）	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_bldg_3dtiles_[行政コード]_[区名]_[lod].Zip	14130_kawasaki-shi_city_2022 citygml_1_op_bldg_3dtiles_14131_kawasaki-ku_lod1.Zip	FME

CMSへアップロードするデータの命名規則（2/2）

データ分類	データ	命名規則	記載例	CMSへのアップロード主体
都市モデルデータ	地物型データ（土地利用モデル）	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_luse_mvt.Zip	14130_kawasaki-shi_city_2023_1_op_luse_mvt.Zip	FME
	地物型データ（洪水浸水想定区域モデル）	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_fld_natl_[水系]_[河川名]_3dtiles_[l1 or l2].Zip	14130_kawasaki-shi_city_2023_citygml_1_op_fld_natl_tamagawa_tamagawa-asakawa_3dtiles_l1.Zip	FME
	ファイル別最大LOD リスト	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_[地物型名]_maxLod.csv	14130_kawasaki-shi_city_2023_citygml_1_op_bldg_maxLod.csv	FME
	品質検査結果	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_[地物型名]_qc_result.Zip	14130_kawasaki-shi_city_2023_citygml_1_op_bldg_qc_result.Zip	FME
ユースケースデータ	ユースケースデータ	uc_[UC番号]_[市区町村コード]_[市区町村名英名]_[提供事業者名]_[整備年度]_[データ名].[データの拡張子]	uc_11_22130_hamamatsu-shi_acn_2023_shimizunoyaike.json	ユーザー
関連データセット	関連データセット	[市区町村コード]_[市区町村名英名]_[提供事業者名]_[整備年度]_[landmark,shelterなど].[データの拡張子]	42202_sasebo-shi_city_2023_landmark.czml	ユーザー
G空間情報センター向けデータ	都市モデルデータ	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション].Zip	14130_kawasaki-shi_city_2023_citygml_1_op.Zip	CMS
	3D Tiles, MVT	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_3dtiles-mvt_[更新回数]_[バージョン].Zip	13100_tokyo23-ku_city_2023_3dtiles-mvt_1_2_op.Zip	CMS
	ユースケースデータ	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_uc[UC番号]_[バージョン].Zip	13100_tokyo23-ku_acn_2023_UC22-1.Zip	CMS
	オルソ画像	[市区町村コード]_[市区町村名英名]_[提供事業者名]_[整備年度]_ortho_[更新回数]_[オプション]_op.Zip	42202_sasebo-shi_[提供事業者名]_2023_ortho_1_op.Zip	CMS

2.1.2 管理者向け機能

（1）アカウント管理機能

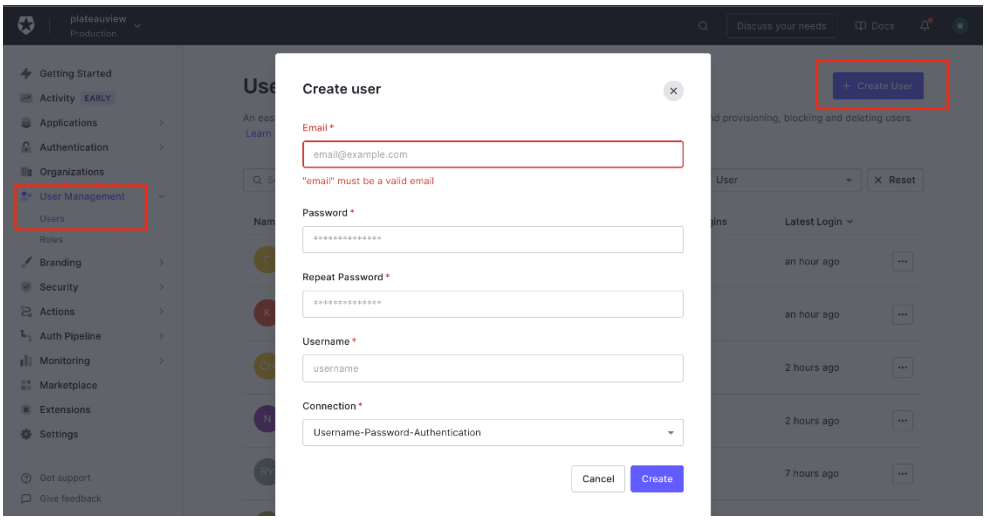
アカウント作成機能

CMSでは、管理者としてCMSを利用するユーザーの管理を行うことができる。
ワークスペースへの追加対象者の情報を準備する。必要となる情報は次のとおり。

- 所属会社
- メールアドレス
- 対象者氏名

CMSの管理者は、Auth0のテナントで、招待するメンバー作成することができる。その後、以下のURLからCMSの利用が可能になる。

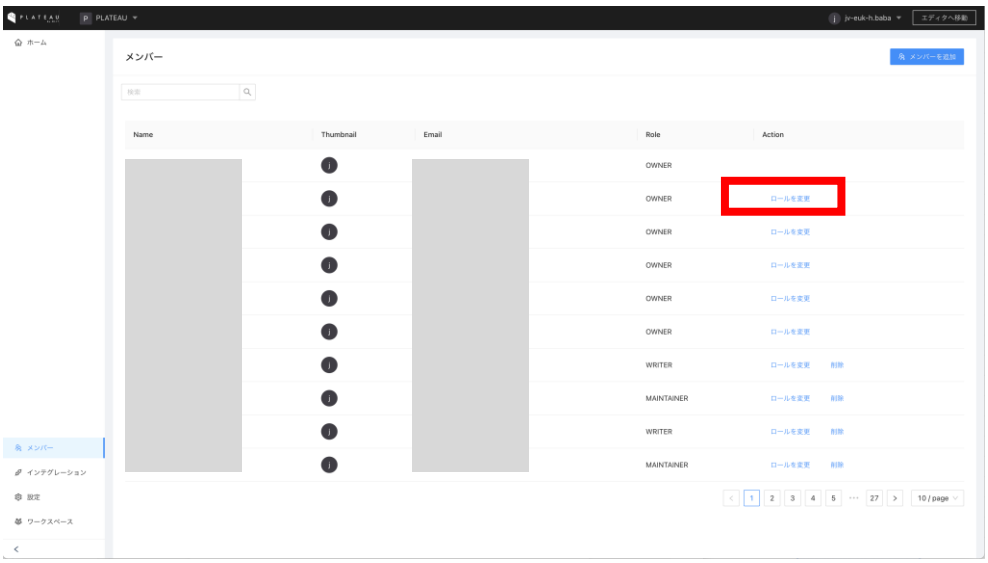
- URL: https://cms.plateauview.mlit.go.jp/
- ID: 個人のメールアドレス
- パスワード: 入力したパスワード



アカウント権限変更機能

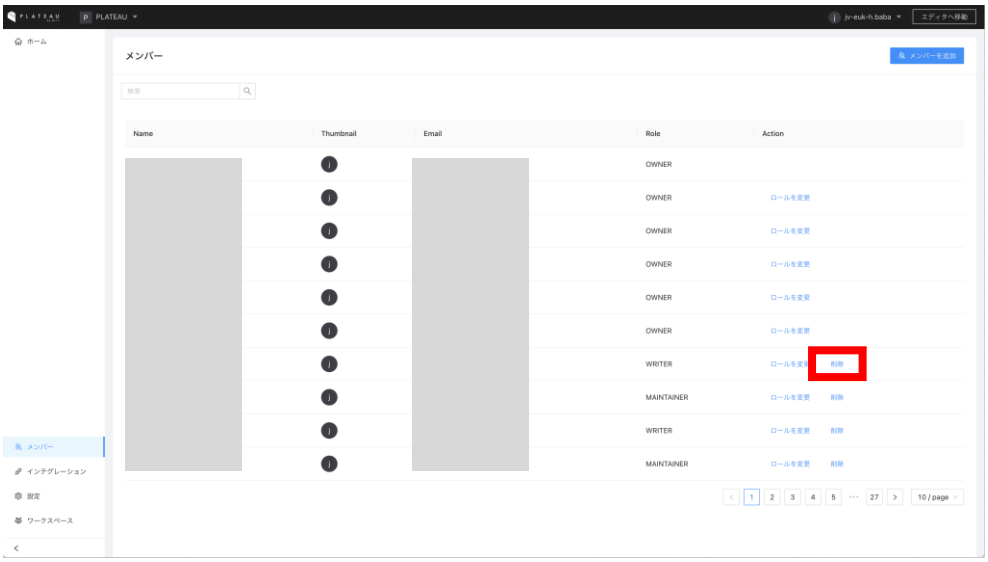
CMSでは、管理者はワークスペース内の権限管理を行うことができる。これによって、多数のユーザーが共同で取り組むプロジェクトにおいても、適切な操作のみをユーザーに許容することができる。権限の種類は以下のとおり。

- オーナー: ワークスペースへのメンバーの招待や削除含めて全ての操作が可能なユーザー
- メインテイナー: ワークスペースへのメンバーの招待や削除以外の操作が可能なユーザー
- 編集者: コンテンツ・アセット・リクエスト・コメントの作成・編集が可能なユーザー
- 閲覧者: 閲覧権限のみ付与されたユーザー



ワークスペースからアカウントの削除

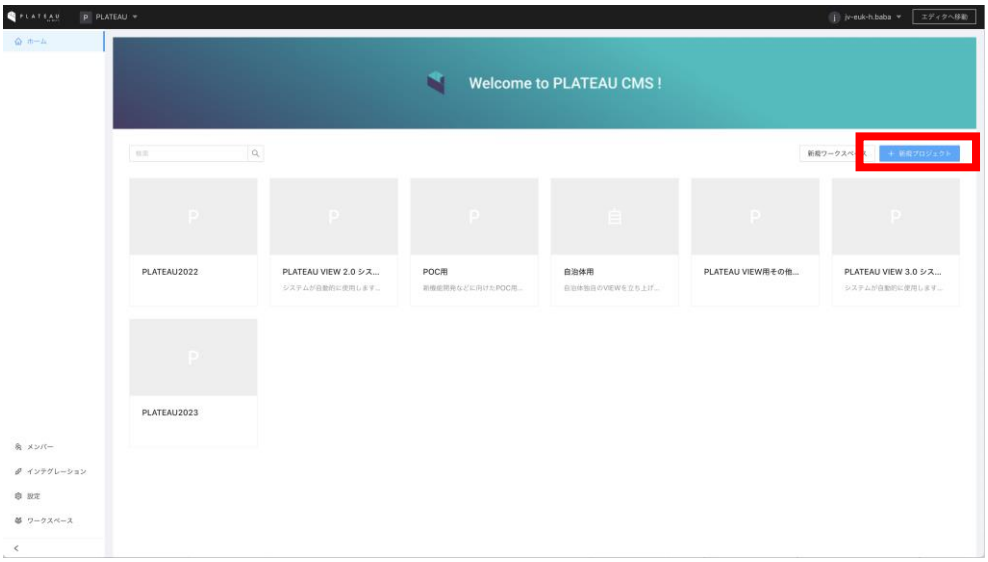
PLATEAU CMSでは、管理者はワークスペース内のユーザーを削除することができる。



（2）プロジェクトの編集機能

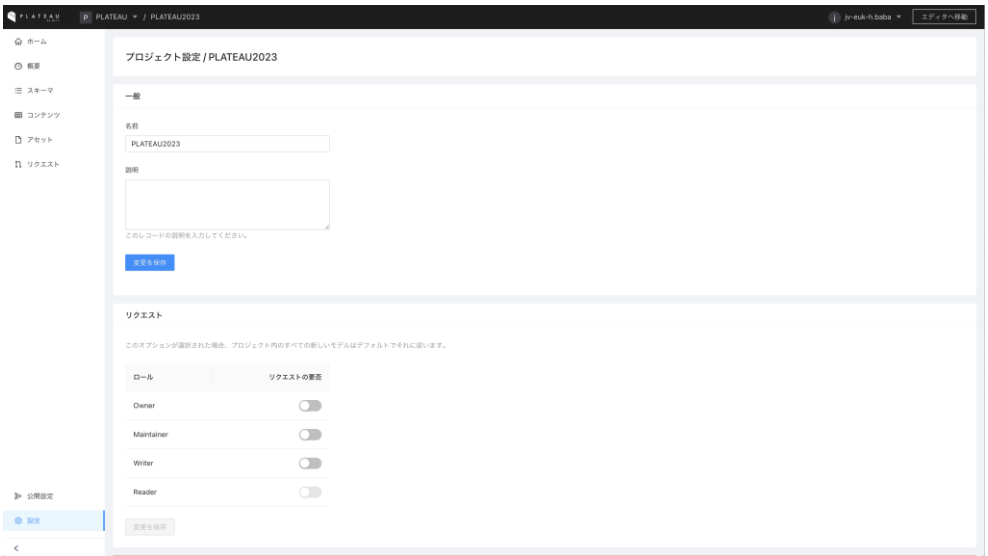
プロジェクトの作成

CMSでは、プロジェクトという単位でデータの管理を行っている。プロジェクトは、任意の目的に応じて管理者が作成・管理することができる。以下の画面では、プロジェクトの一覧表示や新規作成を行うことができる。



プロジェクト設定の変更

プロジェクトの設定ページでは、プロジェクト名や説明の変更ができる。また、リクエストの設定を変更することで、各権限を持ったユーザーがアイテムを直接公開するか、リクエスト機能を経て公開するかを設定できる。リクエスト機能を利用する場合には、レビュワーが承認をすることで、データをCMSから一般公開することができる。

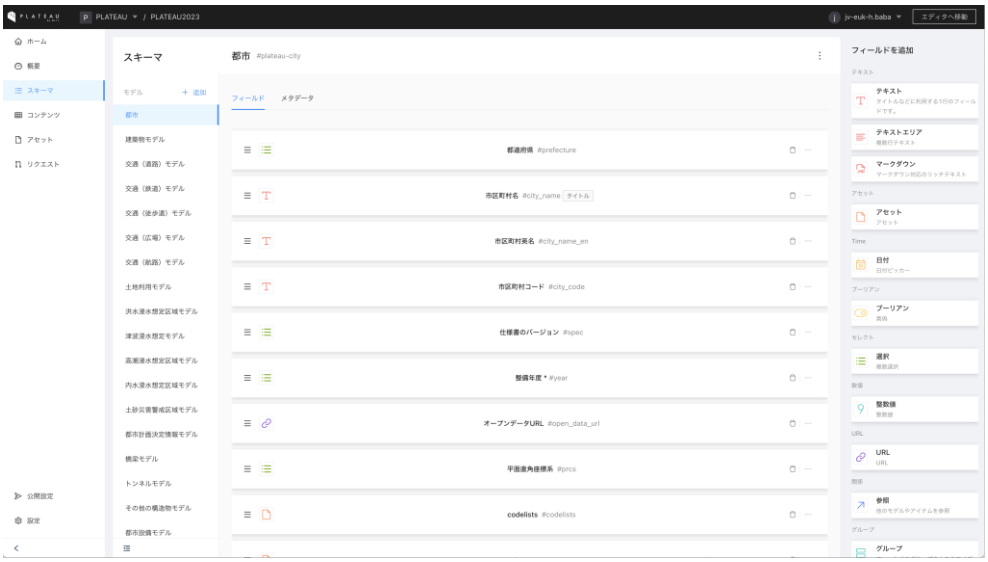


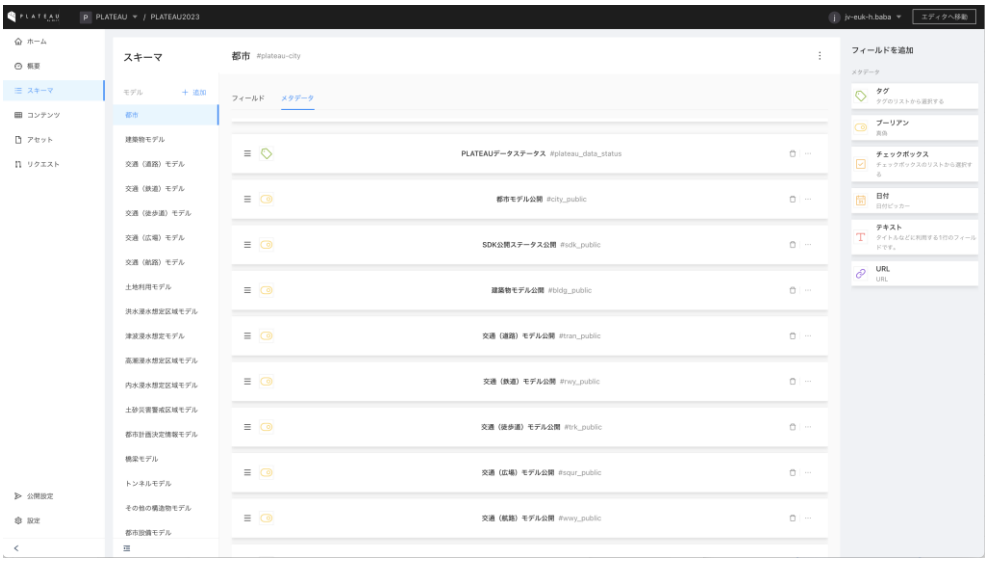
（3）スキーマ編集機能

スキーマの作成

プロジェクト内には、スキーマが存在し、管理者は自由にスキーマを変更することができる。スキーマとは、CMSへ登録するデータ自体のデータ構造を定義するもので、これにより多数のユーザーが共同でデータ登録作業を行なっても、決まった形式でデータを登録することができる。2024年度プロジェクトにおけるスキーマの設定は、3.1.4 PLATEAU CMSの動作確認・セットアップ
で行うため、ここではスキーマ機能の概要を述べる。
スキーマには大きく2種類あり、それぞれ目的に応じて設定をする。2024年度版CMSでは、メタデータにはデータ登録のステータス管理するフィールドなどを設定している。

- フィールド: 登録するデータ自体のデータ構造を定義する。（画像1枚目）
- メタデータ: 登録するデータに対するメタデータ構造を定義する。（画像2枚目）

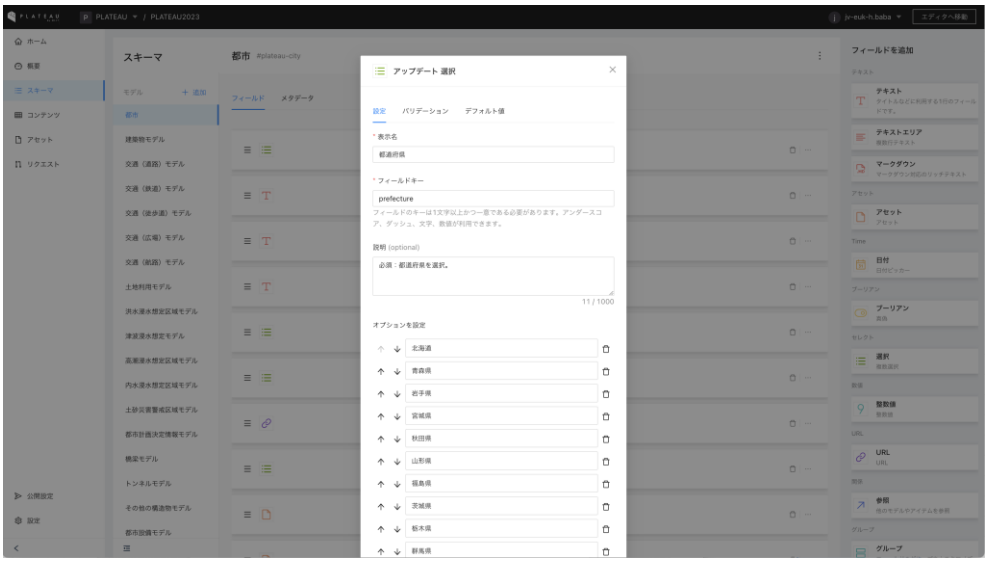




スキーマの変更

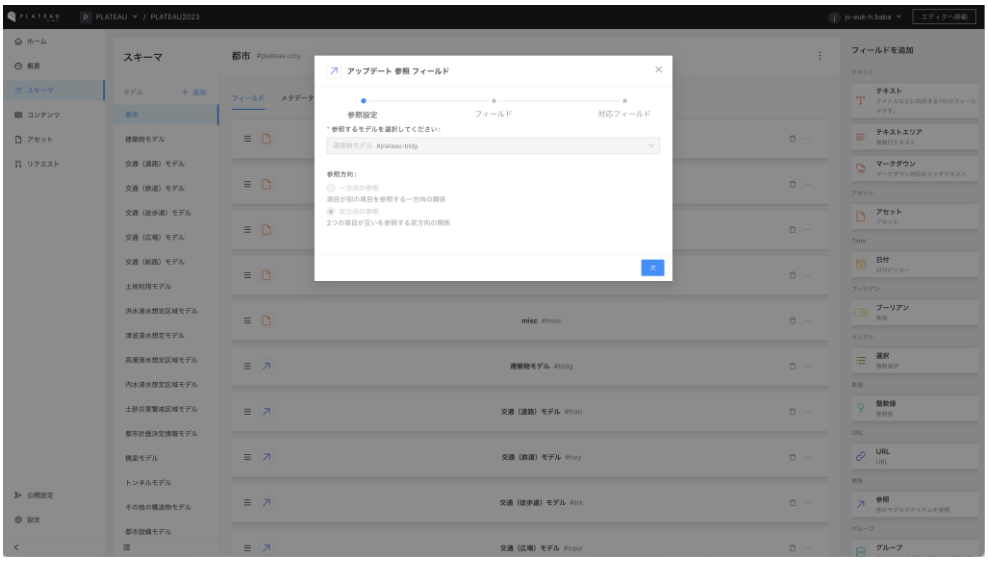
スキーマは後から変更することもできる。スキーマのフィールドの設定変更、フィールドの並び順変更、フィールド自体の削除などが可能である。

また、特殊なフィールドとして、「参照フィールド」と「グループフィールド」が存在する。これらについて、以下で解説する。



参照フィールド

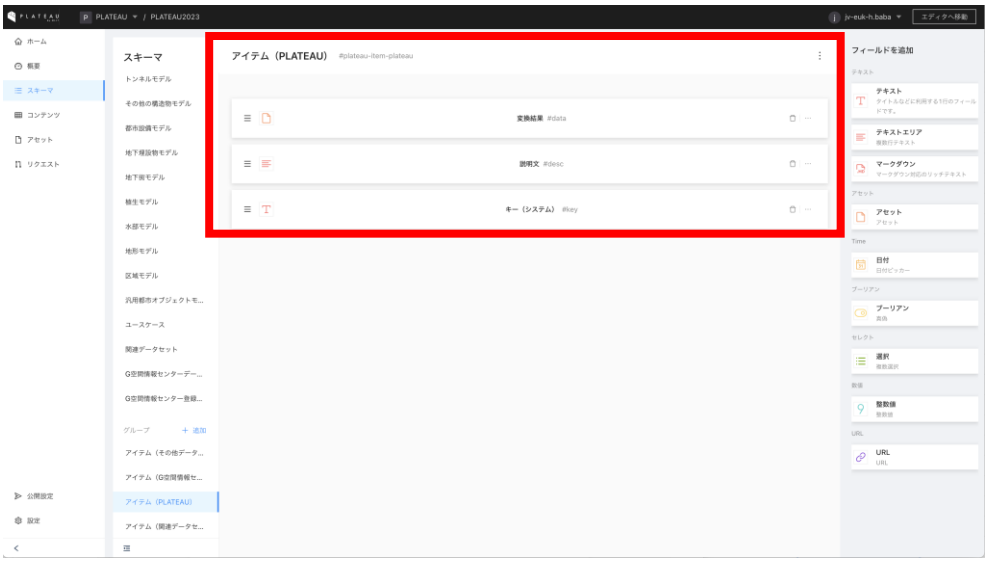
参照フィールドとは、Excelの参照機能のようなもので、モデルを跨いでひも付けをすることができる機能である。例えば、「都市」モデルと「建築物モデル」モデルをひも付けることで、各地物と都市をまとめている。参照フィールドは、以下のように参照先のモデルをスキーマ設定で選択することで設定可能。

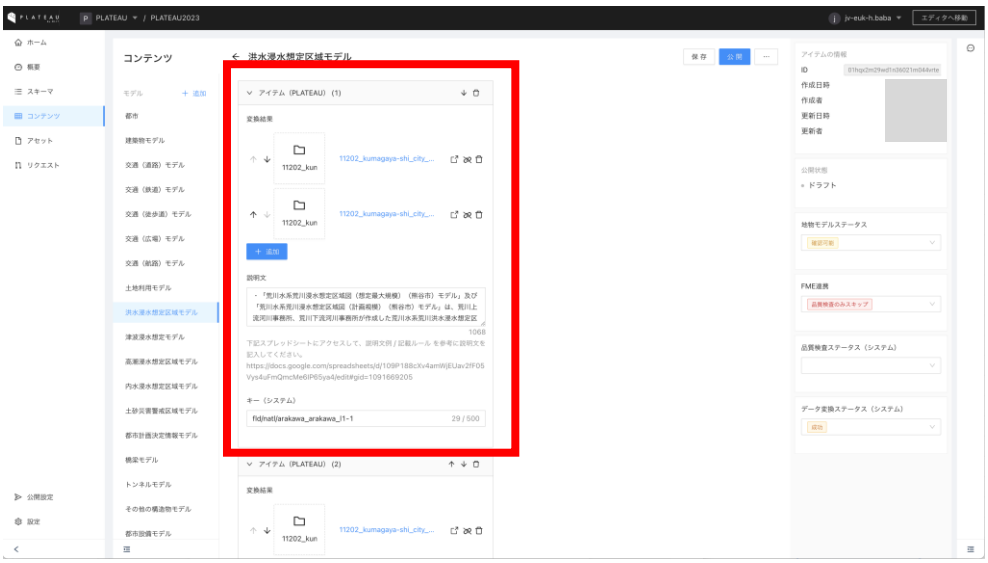


グループフィールド

グループフィールドとは、複数のフィールドを任意の組み合わせでまとめて1つのフィールドとして設定する機能である。データをCMSへ登録する際に、データ本体（アセットフィールド）とそれに付随する説明文（テキストエリアフィールド）を1つの組み合わせとして登録したいケースなどで利用される。CMSでは、洪水浸水想定区域モデルなどにおいて、河川のデータ本体と、各河川の説明文を登録するために利用している。

以下の例では、グループフィールドとして「変換結果（アセットフィールド）」、「説明文（テキストエリアフィールド）」、「キー（システム）（テキストフィールド）」の3つのフィールドを1つのグループとして設定している。下の画像は、これらを合わせたグループフィールドにおけるデータ入力画面の例である。

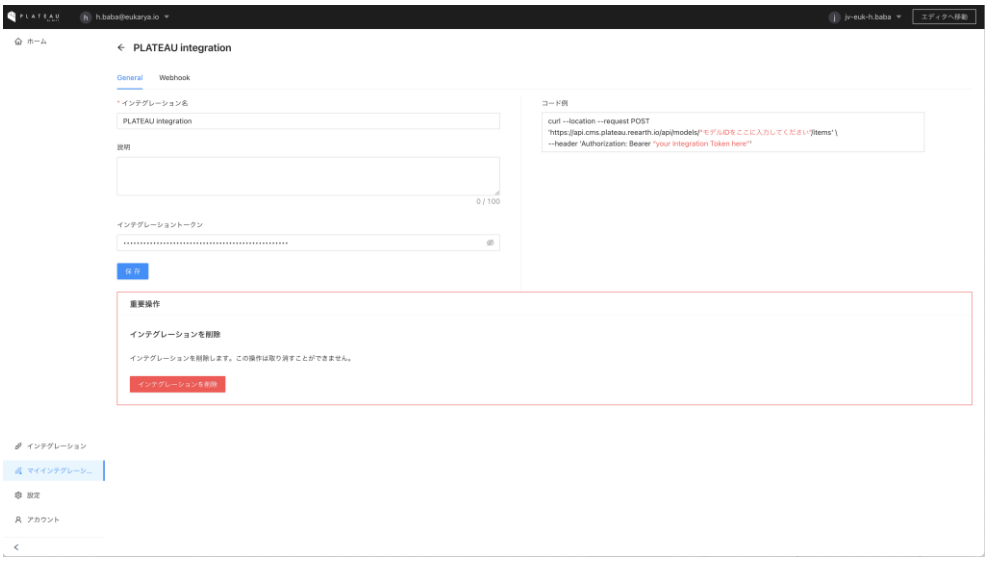




（4）インテグレーション機能

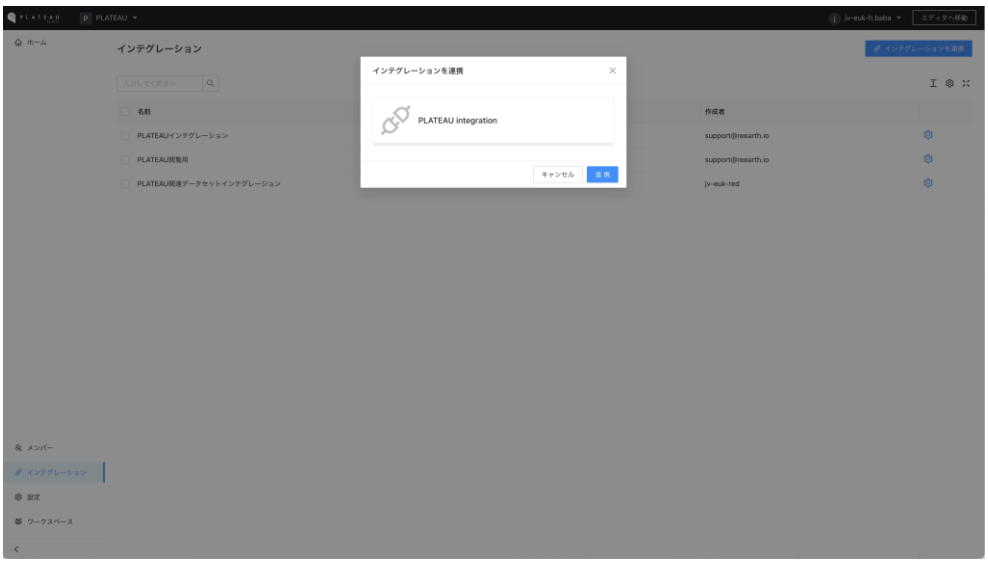
インテグレーションの作成

インテグレーション機能は、CMSが外部サーバーと連携するための仕組みである。同様の事例としては、SlackのApp機能がある。インテグレーションは、管理者が作成と連携をすることができ、APIキーを利用して、通常ユーザーと同様にアイテムの追加、編集、削除等を行うことができる。
PLATEAU CMSでは、このインテグレーション機能を利用して、CMSとFMEの連携を行っている。インテグレーションは個人アカウントにひも付き、そのインテグレーションを利用したいワークスペースへ招待することで利用が可能になる。



インテグレーションの連携

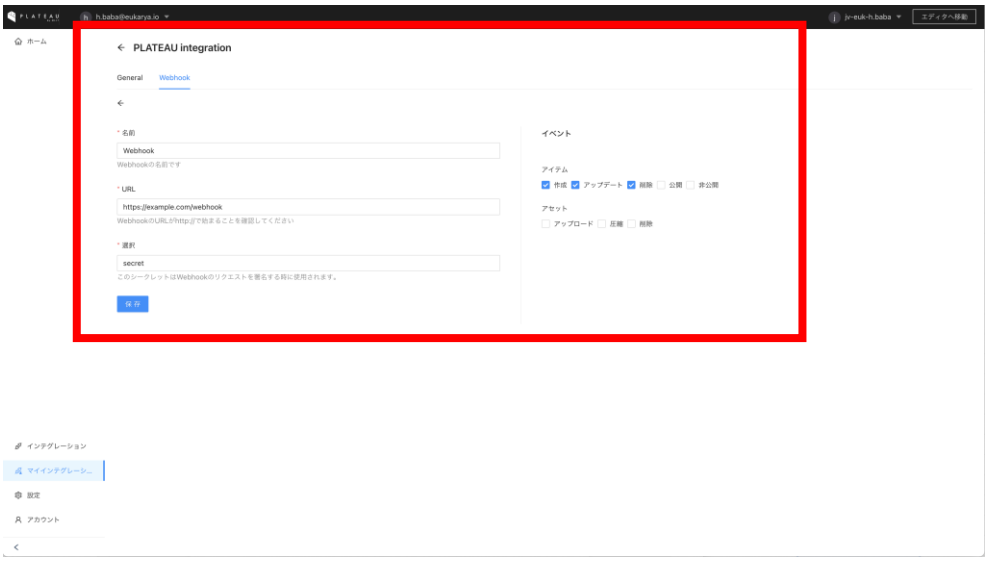
インテグレーションを作成すると、管理者は自身が所属しているワークスペースにインテグレーションを連携することができる。インテグレーションは、通常ユーザーと同様に見なすことができ、通常ユーザーと同様にロールが存在する。適切なロールを割り当てることで、CMSで管理されているデータへの操作をインテグレーションが行うことができる。



（5）Webhook機能

Webhookの設定

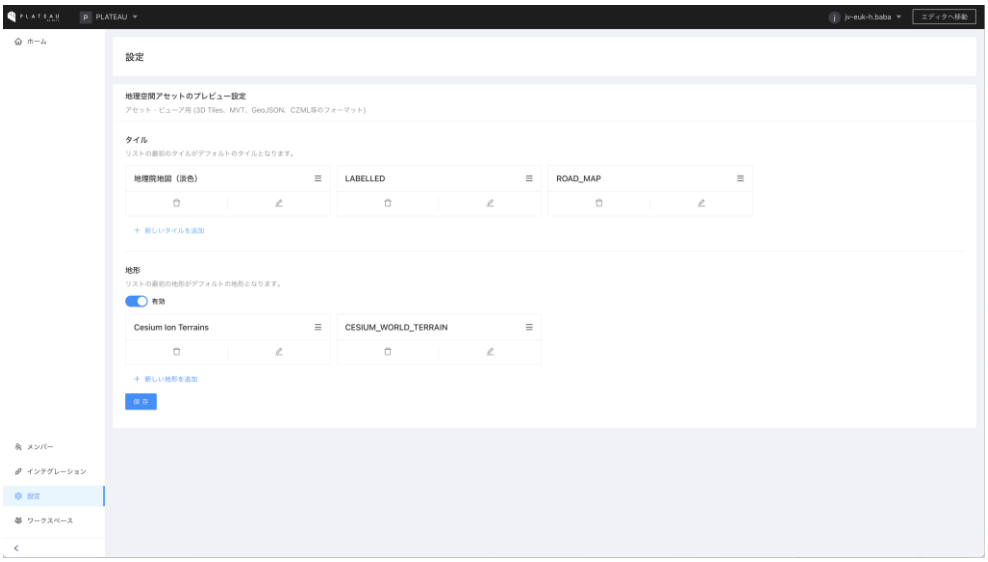
Webhookとは、Webアプリケーションにおいてユーザー定義のHTTPコールバックを設定できる機能である。例えば、「アイテムが新規に登録された時」、「アセットがアップロードされた時」など任意のイベント発生時に、そのイベントを外部アプリケーションへ通知することができる仕組みである。CMSでは、アイテムが更新された際にWebhookをサイドカーサーバーへ通知し、その内容をFMEへ連携することで、3D都市モデルの品質検査やデータ変換を行っている。
以下のように、通知したいイベントと通知先サーバーのエンドポイントを設定する。



（6）アセットプレビュー設定機能

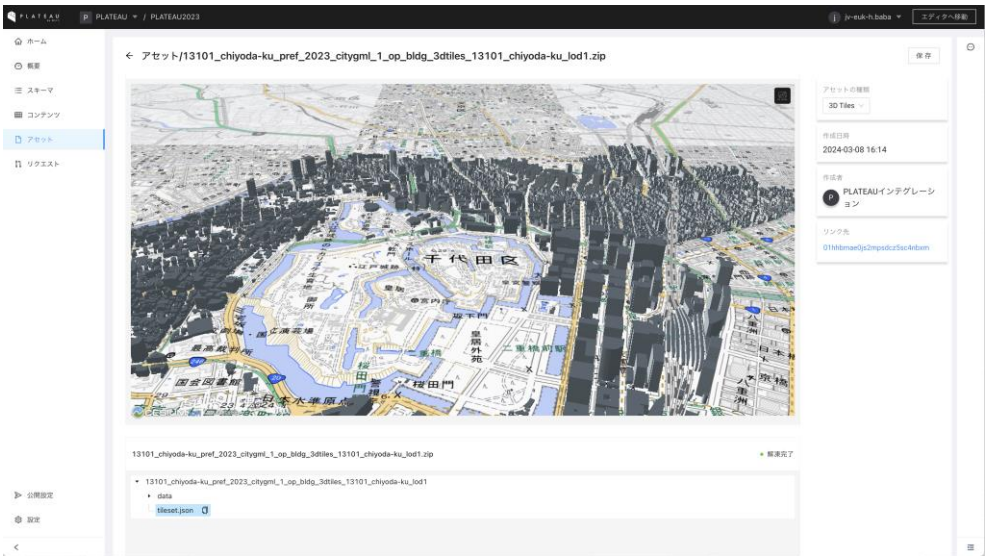
アセットプレビュー設定例

CMSではアセットとしてアップロードされたGISデータの簡易的なプレビュー機能を提供している。管理者はアセットプレビュー設定機能を通して、プレビューで利用する背景地図や地形データの設定を変更することができる。以下の例では、地理院地図（単色）やPLATEAU Terrainを利用するための設定をしている。



アセットプレビュー設定結果

上記のように設定をすると、以下のようにアセットのプレビューをすることができる。地理院地図（単色）を利用することで、閲覧中のエリアなどがわかりやすくなっている。



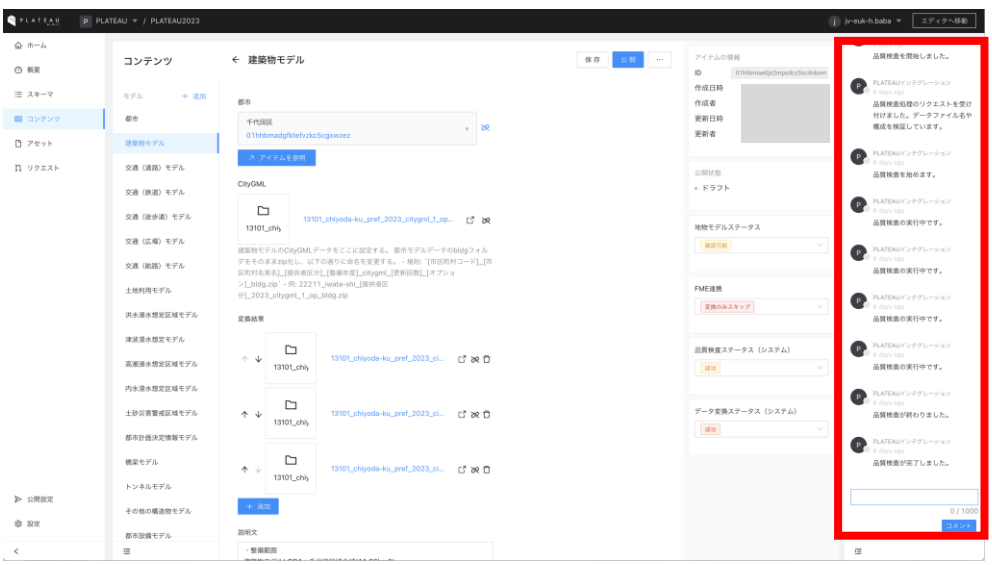
2.1.3 データ登録者向け機能

（1）データ登録機能

データ登録者はスキーマに沿ってデータの登録を行うことができる。定義されたスキーマに沿ってデータを登録後、保存をする。

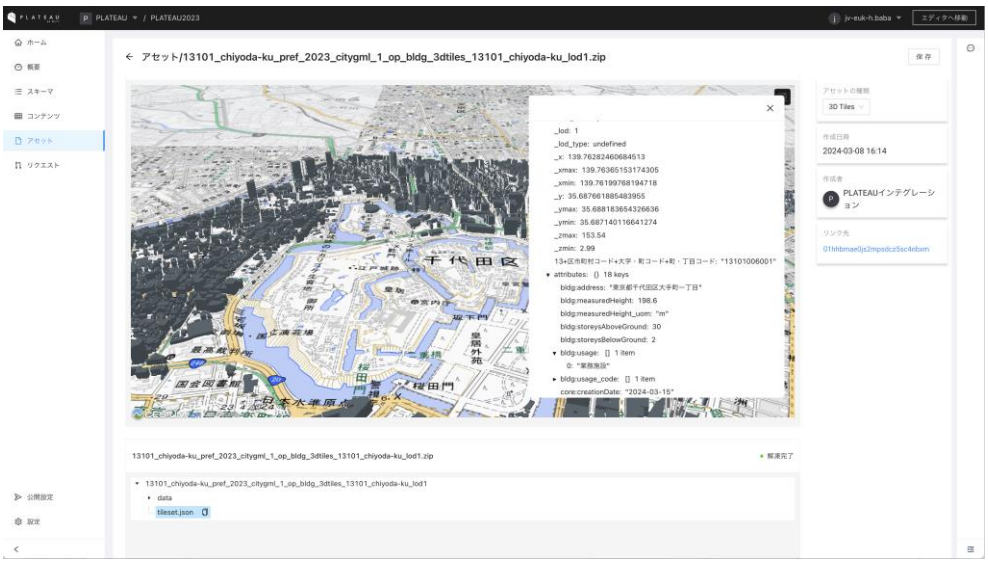
（2）コメント機能

作成したアイテムにはコメントを残すことができる。また、FMEとの連携におけるシステム側からのログもこのコメントに投稿がされる。



（3）アセットプレビュー機能

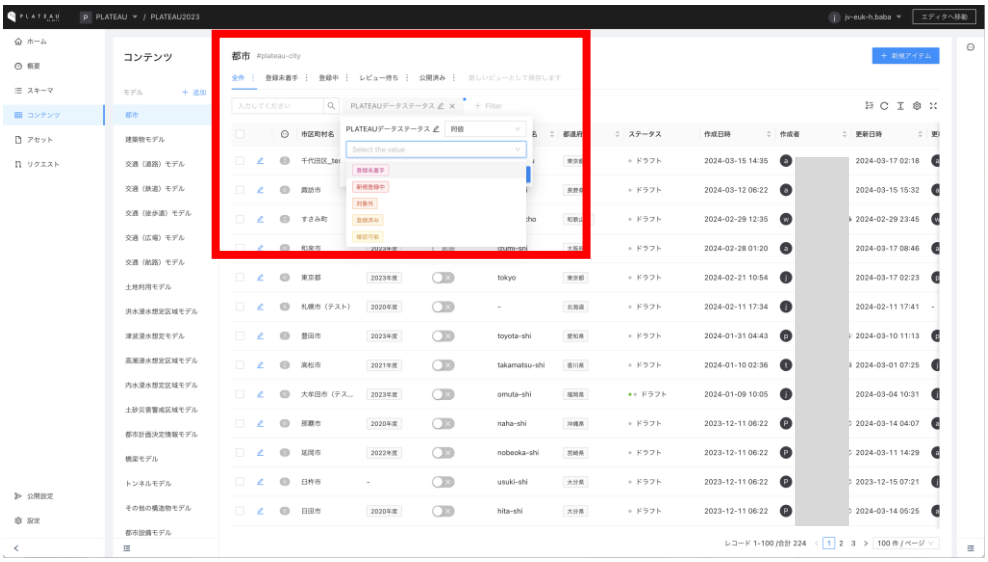
アセットプレビュー機能を利用することで、CMS上でGISデータの簡単な確認をすることができる。地物を押下すると、その地物の属性が表示される。ここに表示される属性は、日本語訳などされていない変換後データが持っている属性である。FMEによるデータ変換が適切に行われているかなどの確認に利用することができる。



（4）ビュー設定機能

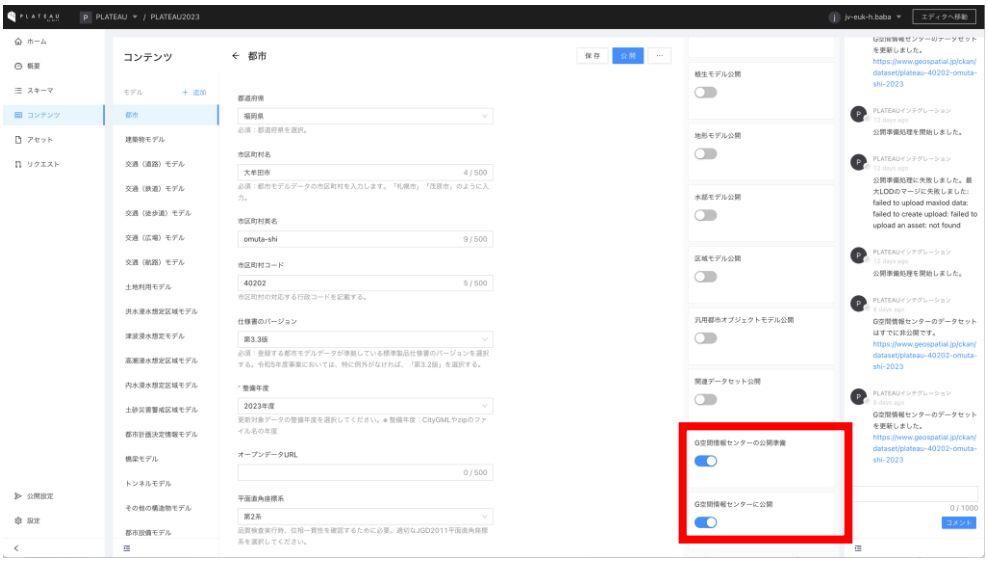
ビュー設定機能とは、アイテムの一覧表示方法をカスタマイズする機能である。類似例はExcelのカスタムビュー機能である。CMSでは、デフォルトでスキーマに定義された順でフィールドを表示する。ビュー機能を利用し、アイテムのフィルター条件や並べ替え条件を設定することで、表示条件を保存し、次回以降の利用時にも同様の表示をすることができる。カスタマイズできる内容は以下のとおり。

- フィルター:アイテムのフィルター条件を設定
- ソート:アイテムの並び変え条件を設定
- カラム:表示するカラムや順序を設定

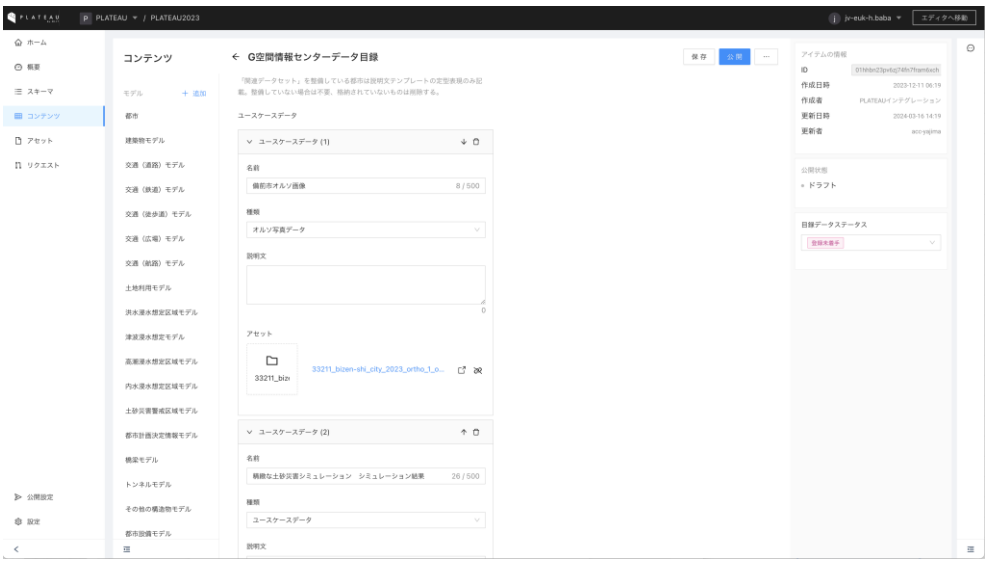


（5）G空間情報センター連携機能

CMSはG空間情報センターとの連携機能を有する。G空間情報センターへの公開設定は「都市」モデルから行う。「G空間情報センターの公開準備」、「G空間情報センターへ公開」をONにすることで、公開処理が実行される。 「G空間情報センターの公開準備」をONにすると、対象都市にひも付けされた地物型ごとのデータをCMSがまとめ、1つのZipファイルを生成する。合計ファイルサイズが大きい場合には、時間がかかることもある。（60GB程度のZipファイルの結合には約4時間程度必要）公開準備が完了し、 「G空間情報センターへ公開」をONにすることで、「G空間情報センターデータ目録」モデルに登録されたテキスト情報と合わせてG空間情報センターへの公開が行われる。なお、デフォルトではG空間情報センターへはプライベートページとして公開されるため、一般公開するにはG空間情報センターで公開設定を変更する必要がある。



また、ユースケースデータやオルソ画像など追加のデータをG空間情報センターへ登録したい場合には、「G空間情報センターデータ目録」モデルにアセットとして登録する。

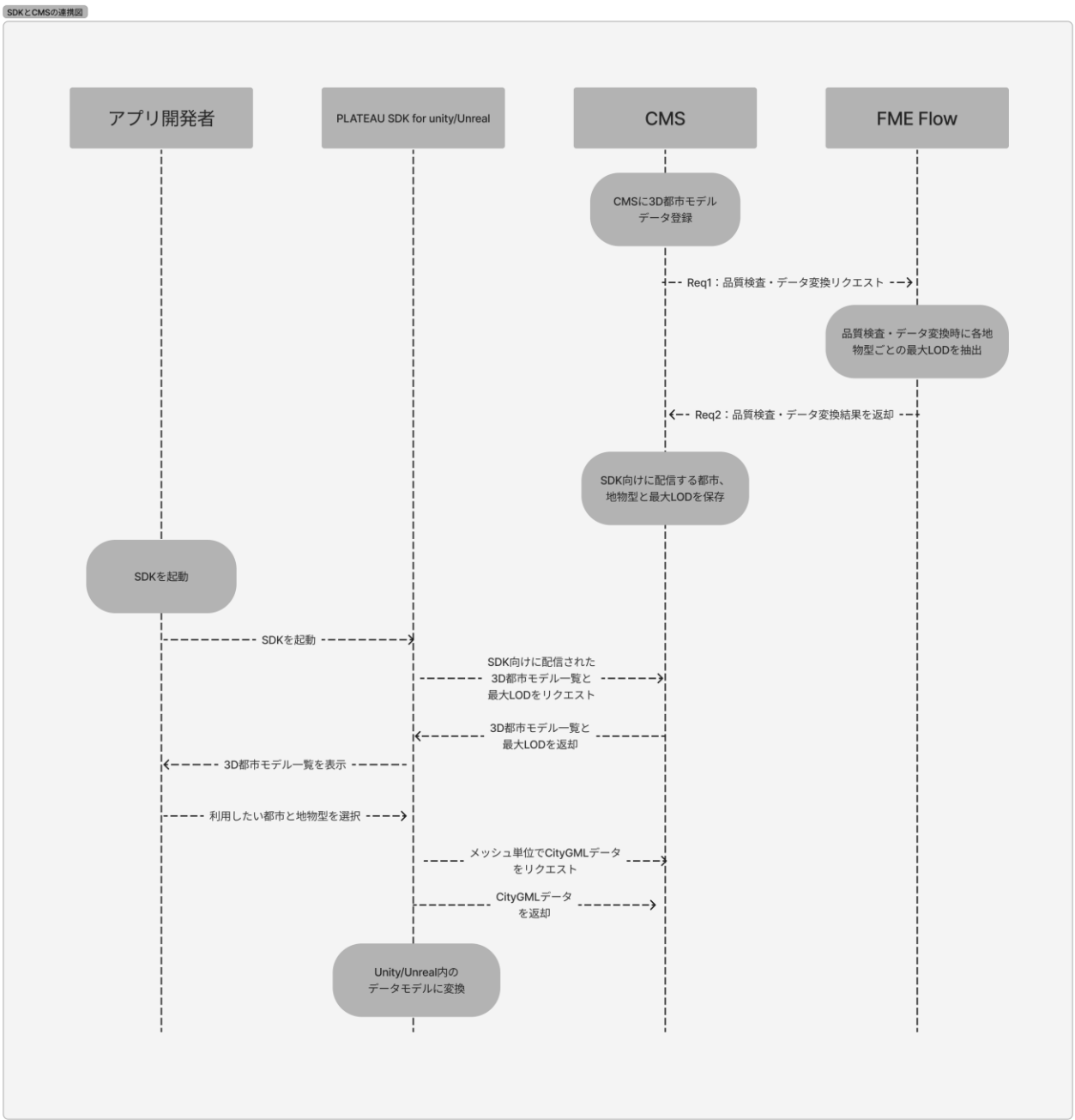


2.1.4 API仕様

（1）PLATEAU SDK for Unity/Unrealとの連携

CMSはいくつかの外部アプリケーションとシステム間連携を行っている。以下に、PLATEAU SDK for Unity/Unrealとの連携仕様を記載する。PLATEAU SDK for Unity/Unrealでは、SDKが3D都市モデルが利用可能な都市、地物型、最大LOD一覧を表示し、アプリ開発者が利用したいデータを選択する方式となっている。これらの連携を実現するために、SDK、CMS、FME Flowでは以下のような連携を行っている。CMSに3D都市モデルデータが登録され、品質検査及びデータ変換を実行する際に、FME FlowがCityGMLファイルから地物型ごとの最大LODを抽出する。抽出された最大LODをCMSに保存し、CMSはSDKから発行される都市、地物型、最大LODの一覧取得をするリクエストに応答する。

図　複数システム連携図



以下にSDKからCMSへ発行されるリクエストと対応するレスポンスを記載する。

（1）共通項目

- Base URL:`https://api.plateau.reearth.io`
- メッシュコードは全て三次メッシュ単位である。
- Access token: （セキュリティの観点から本ドキュメントには記載しない）

全件取得

本エンドポイントでは、SDKから利用可能な都市、地物型が返却される、

```
GET /sdk/datasets
Authorization: Bearer xxx
```

```
{
 "data": [
  {
   "title": "東京都",
   "data": [
    {
     "id": "tokyo-23ku",
     "title": "23区"
     "spec": "2.3", //or "3.0"
     "description": "xxxx", // 建築物モデルの説明文
     "featureTypes": ["bldg", "dem"]
    },
    {
     "id": "hachioji-shi",
     "title": "八王子市",
     "spec": "2.3", //or "3.0"
     "description": "xxxx", // 建築物モデルの説明文
     "featureTypes": ["bldg", "dem"]
    }
   ]
  },
  {
   "title": "神奈川県",
   "data": [
    {
     "id": "yokohama-shi",
     "title": "横浜市",
     "spec": "2.3", //or "3.0"
     "description": "xxxx", // 建築物モデルの説明文
     "featureTypes": ["bldg"]
    }
   ]
  }
 ]
}
```

ID指定してファイル一覧を取得

本エンドポイントでは、対象都市の各地物型ごとの最大LODと3次メッシュ単位のCityGMLファイルへのURLが返却される。SDKはこのCityGMLをダウンロードしている。

```
GET /sdk/datasets/:id/files
Authorization: Bearer xxx
```

```
{
 "bldg": [
  {
   "code": "12345678",
   "url": "https://xxxx/xxxxx/12345678_hogehoge.gml",
   "maxLod": 2
  },
  {
   "code": "12345679",
   "url": "https://xxxx/xxxxx/12345679_hogehoge.gml",
   "maxLod": 2
  }
 ],
 "veg": [
  {
   "code": "123456",
   "url": "https://xxxx/xxxxx/123456_hogehoge.gml",
   "maxLod": 2
  }
 ],
 "...": []
}
```

（2）その他の公開API

CMSでは、SDK向けのAPIだけでなく試験的にGraphQL形式のAPIを公開している。GraphQLとは、サーバー向けに発行するリクエストをより柔軟に構築するためのOSS及びクエリ言語である。実際のAPIはhttps://api.plateau.reearth.io/datacatalog/graphqlから確認が可能で、開発に伴いこちらのページは適宜アップデートされている。こちらのGraphiQLページから最新のドキュメントも閲覧できる。本ページでは、このGraphQLAPIを利用して、CMSへ保存されたデータがどのように取得可能かを例示する。

データセット検索

「searchTokens」に検索文字列を入力することで、検索文字列を名称に含むデータの一覧を返却する。

```
query dataset {
  datasets(input:{searchTokens:["建築物"]}) {
    id
    name
    groups
    items {
      id
      name
      url
    }
  }
}
```

```
{
  "data": {
    "datasets": [
      {
        "id": "d_01101_bldg",
        "name": "建築物モデル（中央区）",
        "groups": null,
        "items": [
          {
            "id": "di_01101_bldg_lod1",
            "name": "LOD1",
            "url": "https://assets.cms.plateau.reearth.io/assets/b8/314602-4b39-4d5f-be2d-
a0b17a3e3c21/01100_sapporo-shi_city_2020_citygml_6_op_bldg_3dtiles_01101_chuo-
ku_lod1/tileset.json"
          }
        ]
      },
    ]
  }
}
```

都市とそのデーター覧取得

「code」に都市の行政コードを入力することで、対象都市に関連するデータの一覧を返却する。

```
query area {
  area(code:"13229") {
    id
    type
    name
    datasets{
      id
      name
    }
  }
}
```

```
{
  "data": {
    "area": {
      "id": "c_13229",
      "type": "CITY",
      "name": "西東京市",
      "datasets": [
        {
          "id": "d_13229_bldg",
          "name": "建築物モデル（西東京市）"
        },
        {
          "id": "d_13229_tran",
          "name": "道路モデル（西東京市）"
        },
        {
          "id": "d_13229_lsld",
          "name": "土砂災害警戒区域モデル（西東京市）"
        },
        {
          "id": "d_13229_urf_UseDistrict",
          "name": "都市計画決定情報モデル 用途地域モデル（西東京市）"
        }
      ]
    }
  }
}
```

2.2 PLATEAU Editor・PLATEAU VIEW

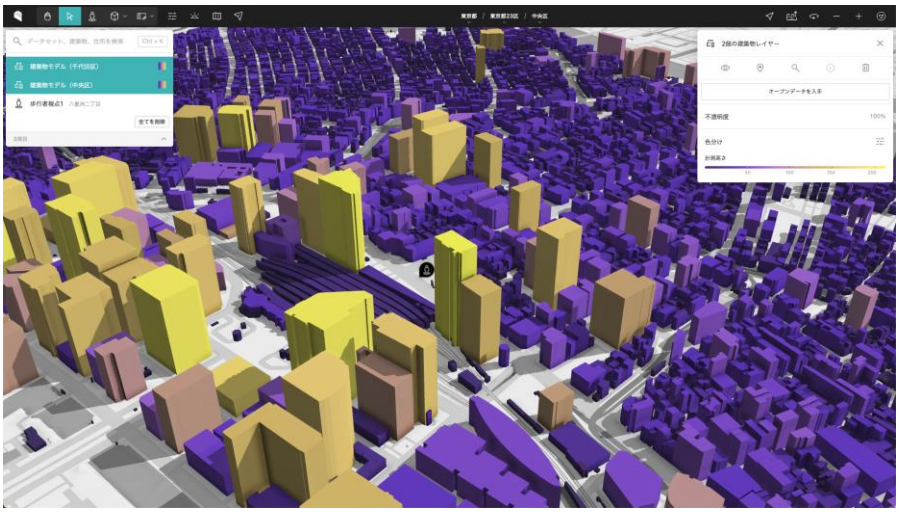
2.2.1 PLATEAU Editor・PLATEAU VIEWとは

1.1のシステム構成で述べたように、PLATEAU VIEW 4.0のシステムの機能は、大きく「データの管理」と「データの可視化」に分かれる。

PLATEAU EditorとPLATEAU VIEWは、このうち「データの可視化」を担うシステムのことを指す。具体的にはこの可視化システムは、更に以下のように分かれる。

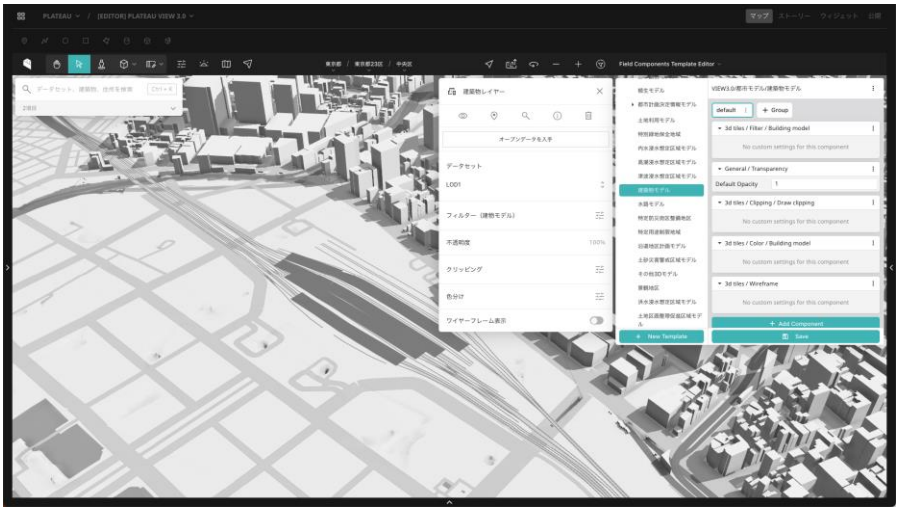
PLATEAU VIEW

- ユーザーが、3D都市モデルデータを初めとするさまざまなGISデータの可視化をWeb上のデジタル3D地球儀上で行うことができる、Webアプリケーション。下記に述べるPLATEAU Editorによって自動的に管理・運用されている。



PLATEAU Editor

- 上記のPLATEAU VIEWそのものを作成及び公開するための、Webアプリケーション。目的に応じてさまざまなWebアプリケーションを作成・公開することが可能であるが、PLATEAU VIEW 4.0では、1つのWebアプリケーションをPLATEAU Editor経由で一般向けに公開し、それをPLATEAU VIEWと呼称している。



（1）使用ソフトウェア・サービス

上記システムを構築するために、オープンソースソフトウェア（OSS）と、有償のクラウドサービスを組み合わせて利用している。クラウドサービスの詳細については次項の各コンポーネントの説明で述べる。

表　使用ソフトウェア・サービス一覧

項目	項目	説明
Re:Earth	OSS	Eukarya社が開発・公開しているOSS。CesiumJSを内包しており、ノーコードで地図やデジタル地球儀を使用したWebアプリケーションを作成・公開することができるWebアプリケーション。プラグインによる機能拡張にも対応。
Cesium Ion	有償クラウドサービス	Cesium社が提供するクラウドサービス。地球儀上の日本における地表面を表現する3Dのデータ（テラインデータ）を配信するために使用している。
Auth0	有償クラウドサービス	Auth0社が提供するクラウドサービス。アカウントの管理・認証・認可を行うIDプロバイダを提供する。Auth0を使用することで、開発者はアプリケーションに安全で使いやすく信頼性の高い認証・認可機能を組み込むことができる。PLATEAU EditorもAuth0を使用して認証・認可機能を実現している。
Google Cloud	有償クラウドサービス	Google社が提供するクラウドコンピューティングプラットフォーム。PLATEAU Editorを動作させるためのサーバーや、ファイルを保存するためのサーバーをGoogle Cloud上に構築している。
MongoDB Atlas	有償クラウドサービス	MongoDB社が提供するクラウドサービス。保守運用が自動化されたマネージドなMongoDBを提供している。MongoDBとは、ドキュメント指向のNoSQLデータベースで、データの柔軟性と拡張性が特徴。PLATEAU CMSはデータベースとしてMongoDBを使用している。
Google Street View	有償API	Google社が提供するGoogle Maps及びGoogle Earthで提供される技術で、ある地点における、パノラマ画像を閲覧することができるサービスである。PLATEAU VIEWでは、この機能を別アプリケーションから利用するための、Street View Static APIを利用している。

（2）使用技術・ライブラリ

PLATEAU Editorはその他さまざまな技術を組み合わせて構築されている。

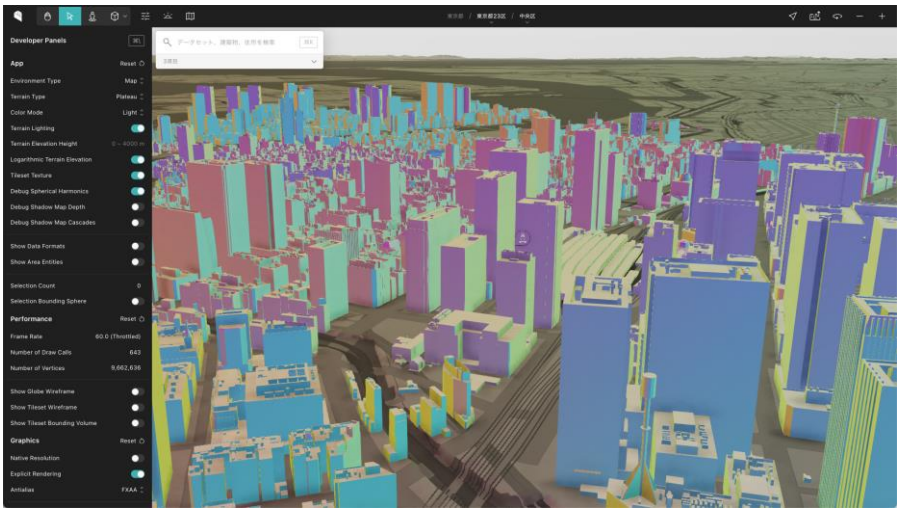
PLATEAU EditorとPLATEAU VIEWでは共通のソースコードが多く存在するが、地図のレンダリングに関連するものは、PLATEAU VIEWの技術として紹介する。

PLATEAU Editorで内部的に利用されている技術・ライブラリ等

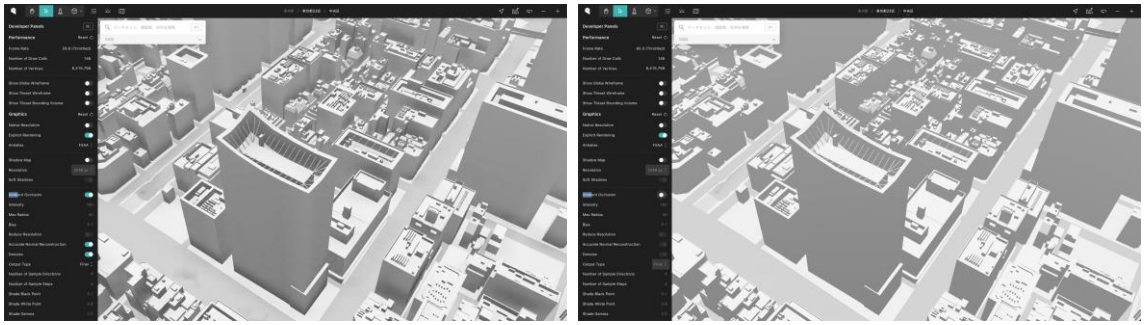
項目	説明
Go	プログラミング言語の1つで、Googleによって開発された。構文がシンプルであり、かつ処理が高速な言語で、主にバックエンド開発に用いられる。PLATEAU Editorのサーバー実装に利用されている。
TypeScript	プログラミング言語の1つで、Microsoftによって開発された。JavaScriptに静的型付けを加えたスーパーセットであり、大規模システムの開発に用いられる。PLATEAU Editorではフロントエンド向け開発に利用されている。
React	Meta社によって開発されたUI構築のためのJavaScriptライブラリ。特に大規模かつ複雑なUI実装において利用される。PLATEAU Editorではフロントエンド向け開発に利用されている。
GraphQL	API向けに作られたクエリ言語及びランタイムを指す。WebAPIの開発において、RESTなどの方式と比較して、より柔軟かつ効率的なAPIの提供を可能にする。PLATEAU Editorではバックエンドとフロントエンド間の通信に利用されている。
Ant Design	Alibaba社が開発したUIフレームワーク及びUIライブラリ。これによりPLATEAU Editorでは統一的なデザインを提供している。
Terraform	HashiCorp社によって開発された、infrastructure-as-codeを実現するためのソフトウェアであり、サーバー構成をコードとして宣言的に管理をできるようにする。
WebAssembly	モダンなブラウザで動作するプログラミング言語の一種。Rustなどその他のプログラミング言語からコンパイルされ、ブラウザ上の隔離環境でプログラムを実行することで、安全で高速に実行される。PLATEAU Editorでは、プラグインシステムなどの動作に利用されている。

（3）使用3DCG技術

球面調和関数
球面上の関数を表すのに用いられる数学的関数。3Dグラフィックスでは、光の反射や放射など、環境内の光の複雑な挙動を効率的に近似し、シミュレートするために使用される。この技術により、リアルタイムレンダリングにおける計算負荷を減少させつつ、高品質なビジュアルエフェクトを実現できる。
以下の画像は、球面調和関数の適用がわかりやすいよう、上下前後左右から別の色を当てている。建築物の各面にグラデーションのような色が適用されているように、環境における光の反射を再現しつつ、計算負荷の最適化を行っている。

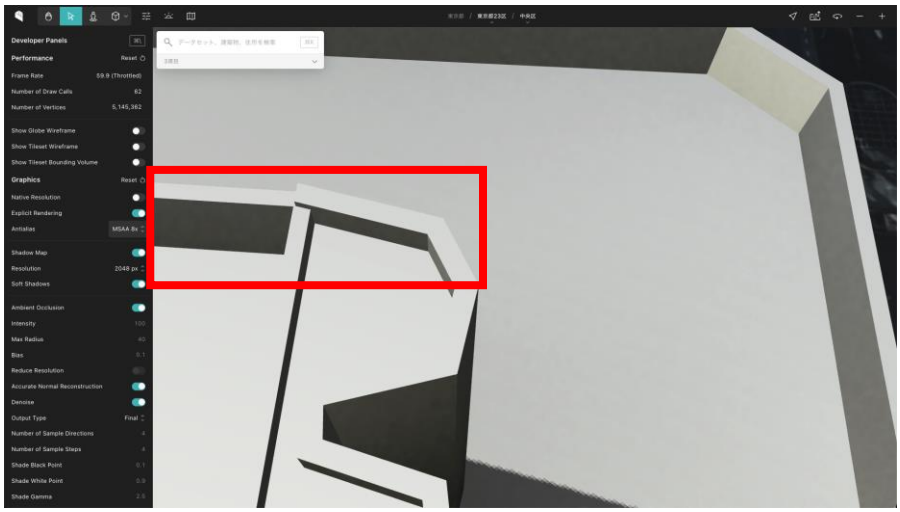


アンビエントオクルージョン
シーン内の各点での局所的な環境光の遮蔽を計算するレンダリング技術。物体の接近する部分や隅など、光が届きにくい場所に陰影を加えることで、よりリアルな3Dシーンを生成する。アンビエントオクルージョンの代表的なものには、SSAO（Screen Space Ambient Occlusion）やHBAO（Horizon-Based Ambient Occlusion）などがある。
以下の画像（左：アンビエントオクルージョンあり、右：アンビエントオクルージョンなし）を比較すると、アンビエントオクルージョンを適用することで、より立体的な視覚表現ができることが分かる。



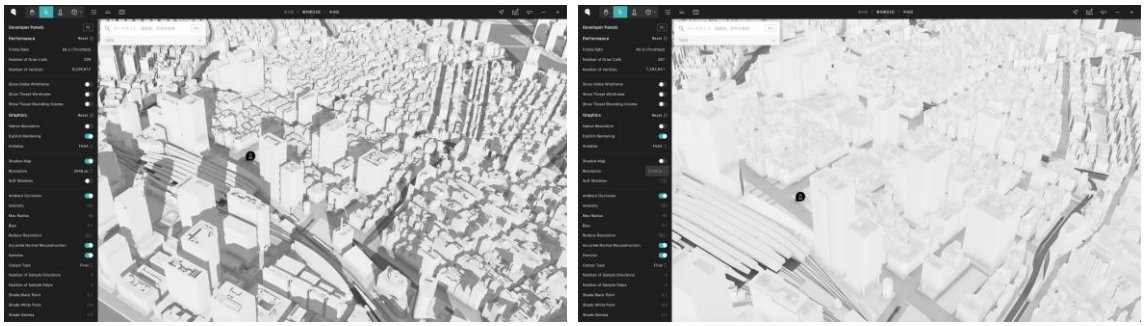
アンチエイリアス
3DCGにおいて、特にエッジ部分に現れるジャギー（エイリアス）をなめらかにする手法。ピクセルの色や輝度を隣接ピクセルとの間で調整し、エッジの滑らかさを向上させることでジャギーを改善する。アンチエイリアスにはいくつか種類がある。
- MSAA（Multi-sample Anti-aliasing）:複数のサンプルポイントを持つピクセルにおいて、エッジ周辺の色を平均化する。
- FXAA（Fast Approximate Anti-aliasing）:ピクセルレベルでエッジを検出し、テクスチャをぼかす。

スクリーンショットではわかりにくいが、アンチエイリアスによって建築物などのエッジのジャギーが緩和されている。



シェーダー
3Dグラフィックスにおいて、光の影響を受けた物体の表面の見え方を決定するためのプログラムまたはコードのセットを指す。シェーダーは、GPU（グラフィックス処理ユニット）上で実行され、リアルタイムで高速な計算を可能にする。PLATEAU VIEW 3.0からは、CesiumJSに加え、独自のシェーダーを実装することで、リッチな立体表現を実現している。
特にメインのシェーダーは以下の2つである。
- 頂点シェーダー:3Dモデルの拡張点に対する操作を行う。頂点の位置、光の影響などの計算に利用する。
- フラグメントシェーダー:画面上の各ピクセルに対する色やテクスチャの計算に利用する。

Shadow mapping
3DCGにおいて影を生成するための手法。光源から見たシーンをレンダリングして、シャドウマップ（深度マップ）を生成し、実際のカメラ位置からシーンをレンダリングする際に、深度マップを参照して、各ピクセルに光が直接照らされているか、影になっているかを判定する。ソフトシャドウを利用して、より自然な影表現をすることも可能。
以下の画像（左：Shadow mappingあり、右： Shadow mappingなし）を比較すると、影表現が適切にされていることが分かる。



（4）PLATEAU Editorの主な機能

PLATEAU Editorでは主に以下の機能が利用可能である。詳しい使い方は、2.2.2を参照されたい。

- ワークスペースの作成・ユーザーの招待
- プロジェクトの作成
- レイヤーの配置・スタイルの設定
- GISデータの読み込み・表示
- シーンの設定変更（ベースマップ・テライン・カメラなどの各種設定）
- インフォボックス（レイヤーの詳細を表示する画面領域）の作成・編集
- ウィジェット（地球儀の上に表示されるさまざまなUI）の配置・編集
- プラグインのインストール（ウィジェットなどを機能拡張可能）
- プロジェクトの公開

（5）PLATEAU VIEWの主な機能

PLATEAU VIEWでは主に以下の機能が利用可能である。詳しい使い方は、2.2.3を参照されたい。

- ヒエラルキーウィンドウ
    - データセットの一覧表示
    - データセット及び住所の検索
    - 地図上へ追加済みレイヤーの表示及び選択
- 凡例の表示・絞り込み表示などの操作・地物の属性表示
- 3D図形の作図
- Google Street Viewとの連携
- 地図の設定の変更（ベースマップの切り替え、地図ラベルの表示）
- タイムラインによる時系列データの表示
- ストーリーテリングの表示と編集
- カメラ操作・現在地の表示
- 統計データ及び標高値によるヒートマップの表示
- 建築物検索機能・建築物クリップ機能
- 共有URLの発行
- ご意見ご要望

2.2.2 PLATEAU Editorの機能

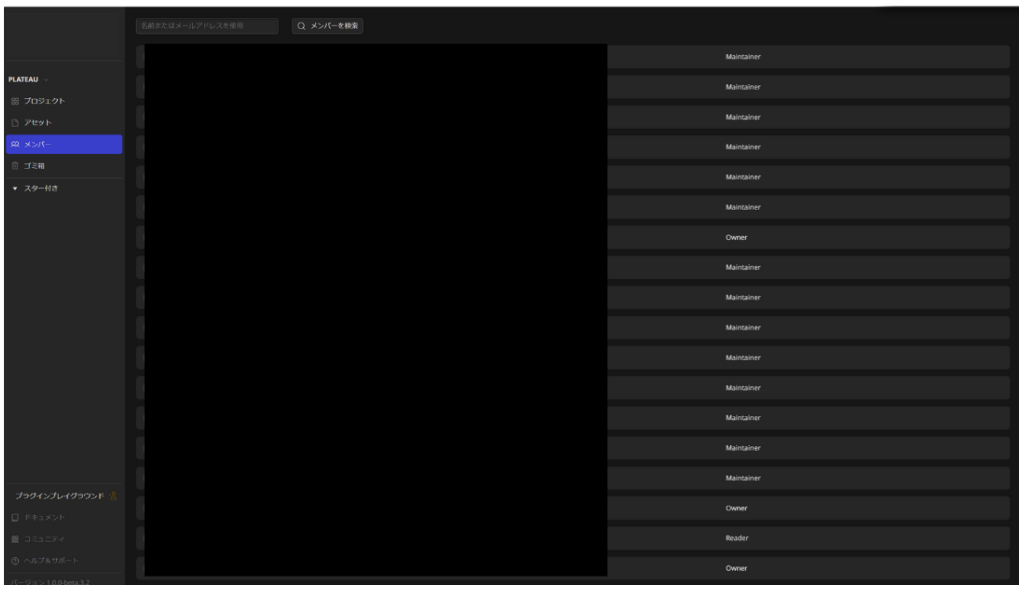
ここではPLATEAU Editorの使用方法について述べながら、PLATEAU VIEW 4.0におけるPLATEAU VIEWをPLATEAU Editor上で作成・公開する手順を解説する。なお、PLATEAU Editorで採用されているOSSであるRe:Earthは非常に多くの機能を持っているが、ここではPLATEAU VIEWを構築するために関係する項目のみを説明する。それ以外のRe:Earthの全ての機能をここで紹介することはできないため、必要に応じて以下のURLも併せて参照されたい。

URL:https://docs.reearth.io/ja/

2.2.2.1 管理者向け機能

（1）アカウント管理機能

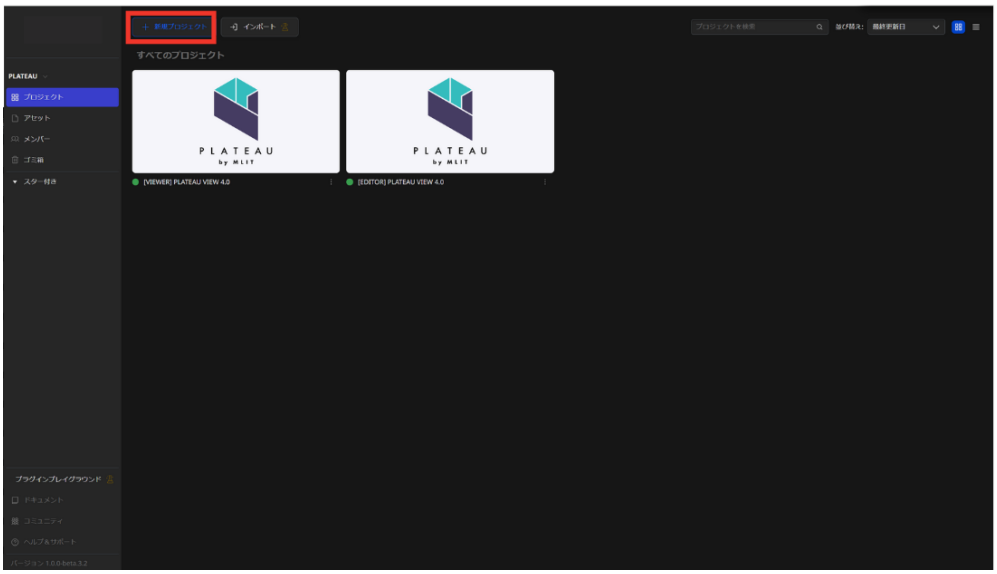
Editorのワークスペースは複数のユーザーが共同でプロジェクトを管理することができる。なお、EditorとCMSではユーザーアカウント及びワークスペースは同じデータベースを参照しており、両者のデータは同期されている。アカウント管理機能の全体像はCMSと同じである。以下のページからワークスペース内のメンバーの権限変更、削除等が可能である。



（2）プロジェクトの編集機能

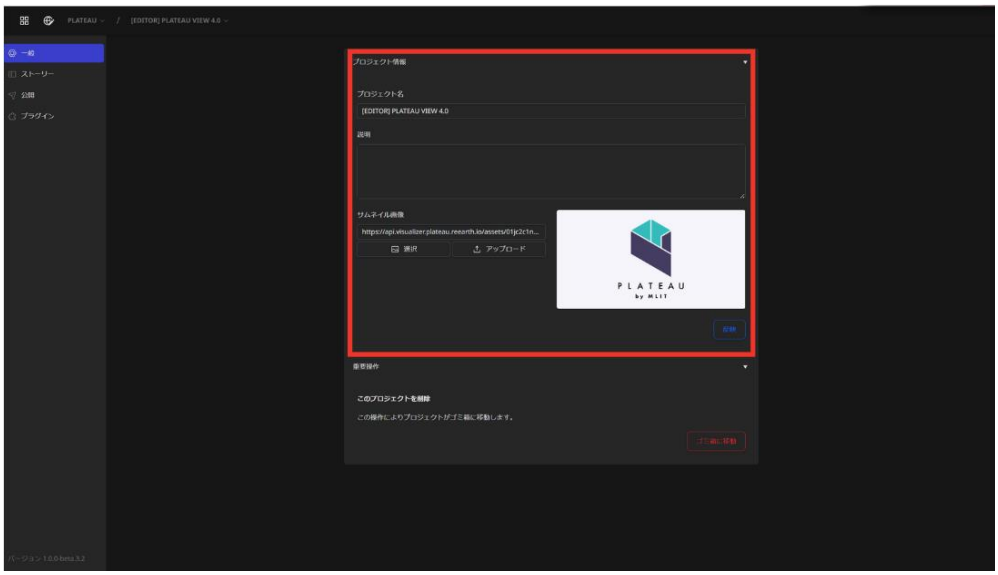
プロジェクト作成機能

Editorでは、CMSと同様にプロジェクトという単位で可視化設定を行っている。プロジェクトは、任意の目的に応じて管理者が作成・管理することができるものである。以下の画面では、プロジェクトの一覧表示や新規作成を行うことができる。

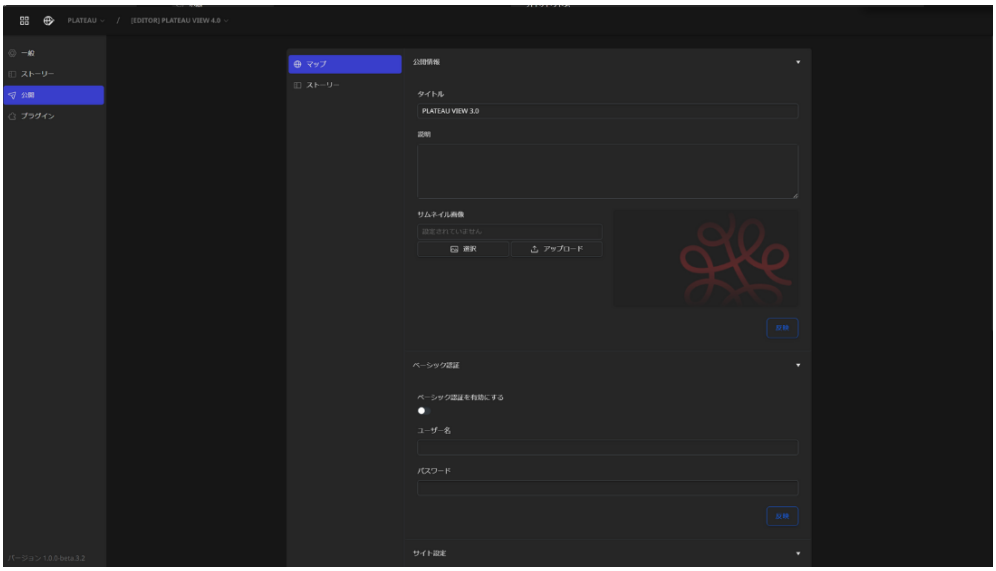


プロジェクト設定変更機能

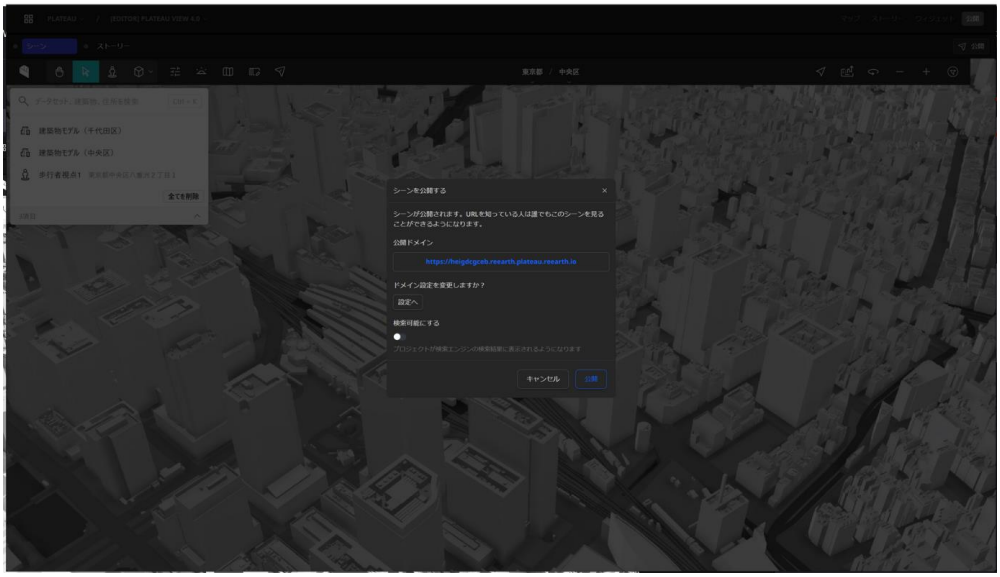
プロジェクトの設定ページでは、プロジェクト名や説明、サムネイルの変更ができる。



プロジェクトの公開設定機能1
プロジェクトの公開設定では、公開後のプロジェクトのページタイトルやベーシック認証の設定等ができる。

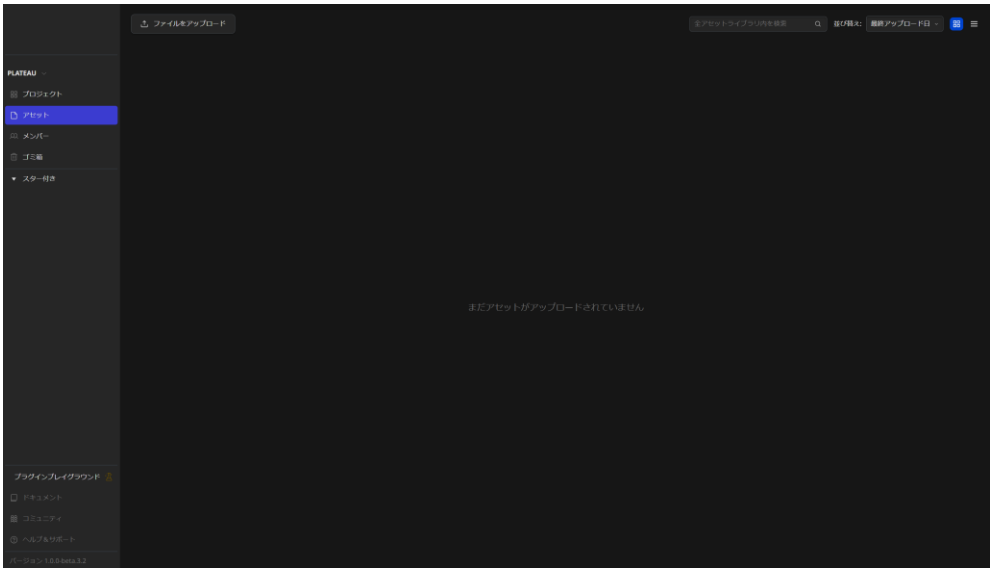


プロジェクトの公開設定機能2
プロジェクトにおいて可視化設定を完了し、公開準備が整ったら、プロジェクト編集ページにから一般公開することができる。一般公開をすると、専用のURLが発行され、URLを通して誰もがプロジェクトの閲覧できるようになる。



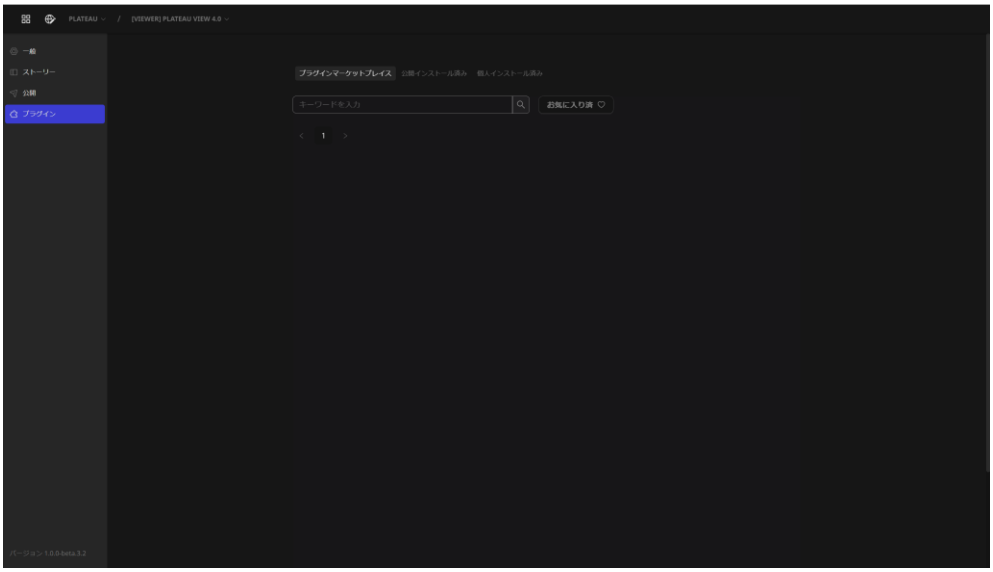
アセット管理機能

アセット管理機能では、Editorにアップロードされたアセットの管理ができる。PLATEAU VIEW 4.0では、3D都市モデルデータをCMSから直接配信しているため、Editorでアセットをアップロードすることは少ないが、以下のようにサムネイル画像などの管理をすることができる。



プラグイン管理機能

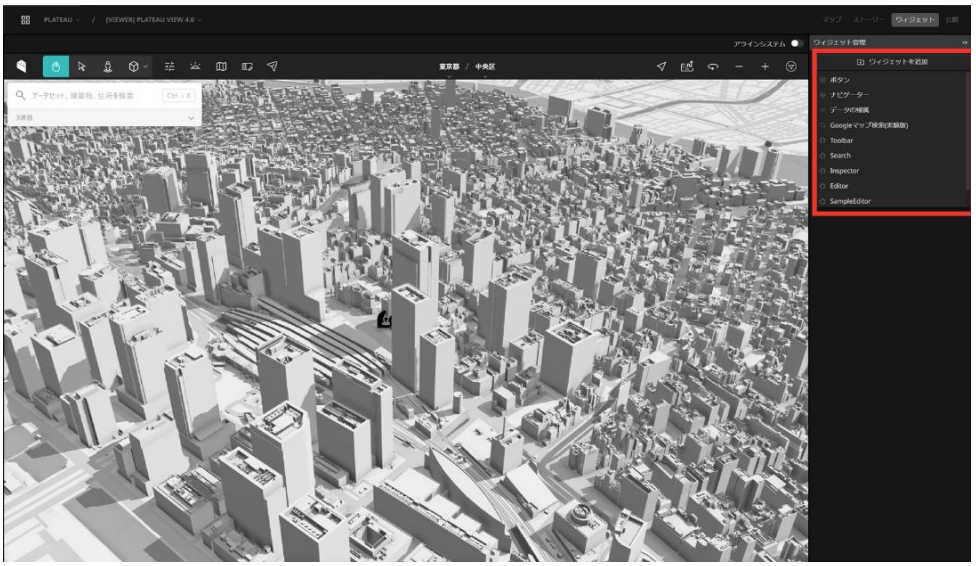
プロジェクトのプラグイン設定では、プロジェクトにインストール済みのプラグインの管理や新規インストールが可能である。プラグインを利用することで、Editor上でデフォルトには存在しない拡張的な機能を利用することができる。利用可能なプラグインの一覧は、Re:Earth Marketplaceで公開されている。



（3）ウィジェット機能

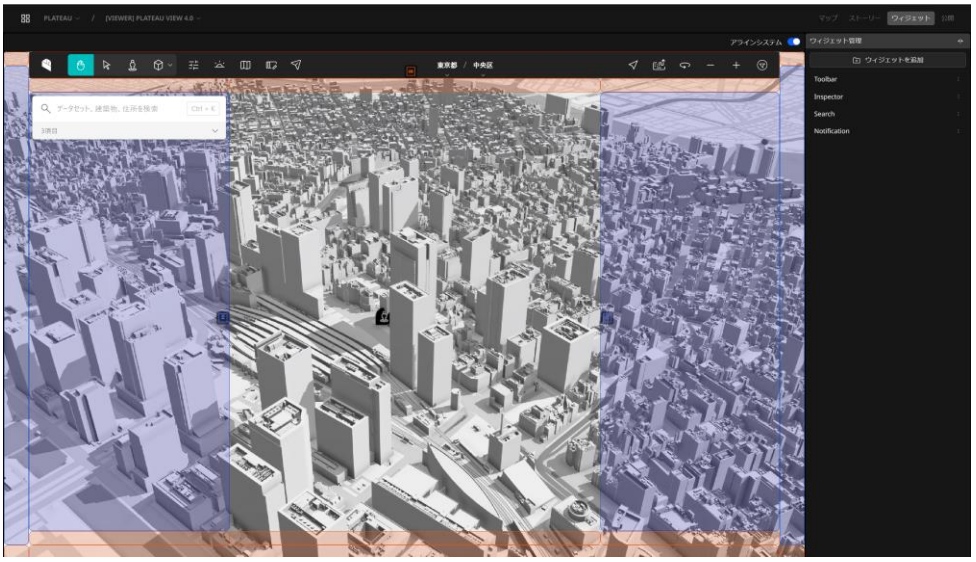
ウィジェット追加機能

Editorではウィジェット（デジタル地球儀とは画面に表示されるUI）の追加や変更をすることができる。



ウィジェット配置変更機能

また、追加したウィジェットの配置を変更することもできる。これにより、UIの自由なカスタマイズが可能である。

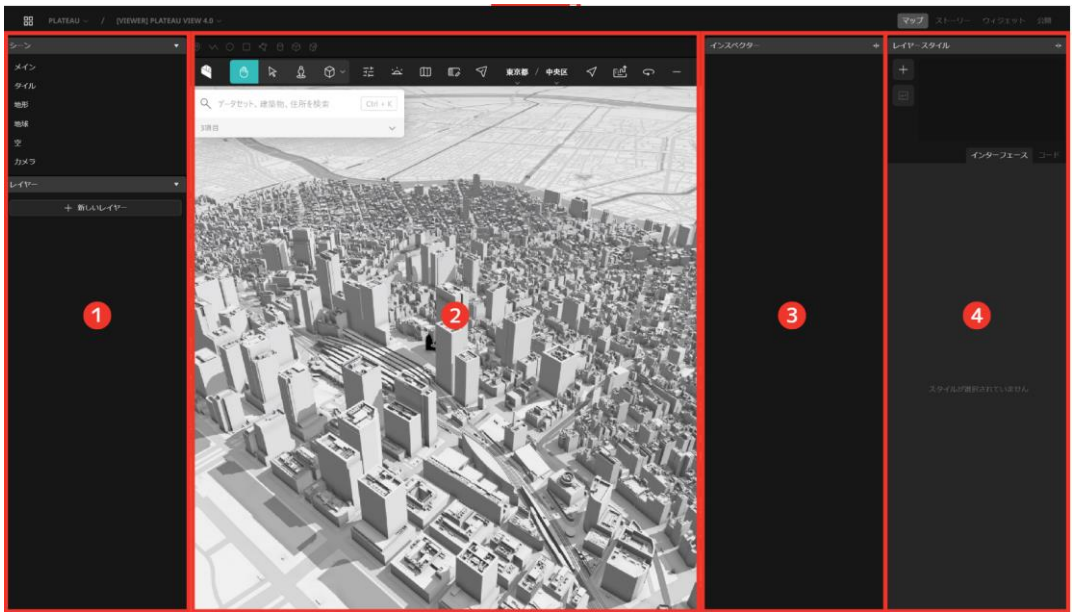


2.2.2.2 データ登録者向け機能

まずEditorーのUIの全体感について説明をする。

1. 左サイドバー
2. 地図Viewer
3. インスペクター
4. レイヤ―スタイル

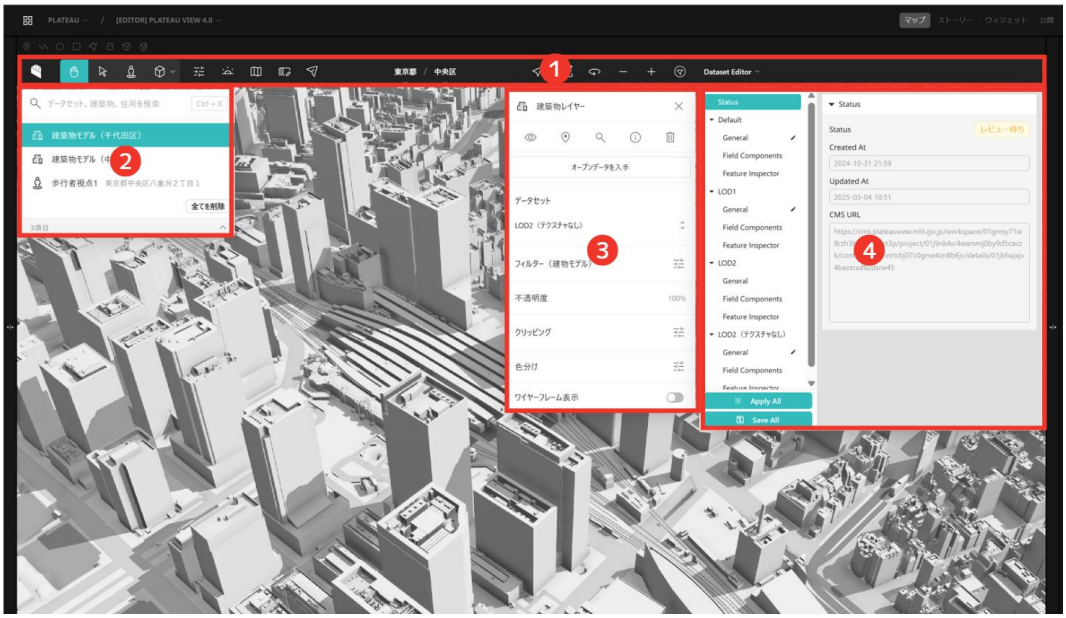
で構成されている。主に2の地図Viewerーを頻繁に利用するため、必要に応じて1、3、4については縮小表示することを推奨する。



各バーの境界をドラッグアンドドロップすることで縮小表示が可能。

地図ViewerにおけるUI

地図Viewerでは、地図上へのデータの追加と、対象データへの色分けや凡例設定等を行うことができる。まずは、地図ViewerにおけるUIのレイアウトを解説する。

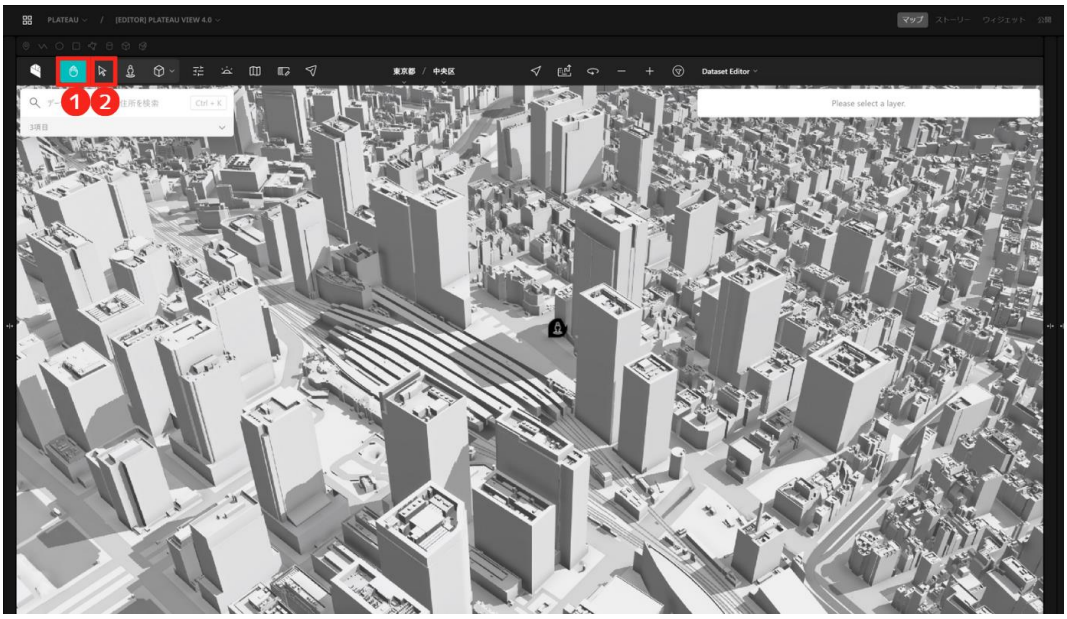


（1）ツールバー

データを閲覧する際のモードの切り替えや、 歩行者モードの利用、作図機能の利用、ベースマップの切り替え等を行うことができる。Editorの設定では基本的には、

1. 移動モード
2. 地物選択モード

の2つのみを利用する。移動モードでは、地図をマウス操作しながら移動することができる。一方で、地物選択モードでは、押下して地物を選択し、地物の属性情報を閲覧することなどが可能。（うまく地物が選択できない場合や、地図を移動できない場合は現在のモードを確認すること）



（2）ヒエラルキーウィンドウ

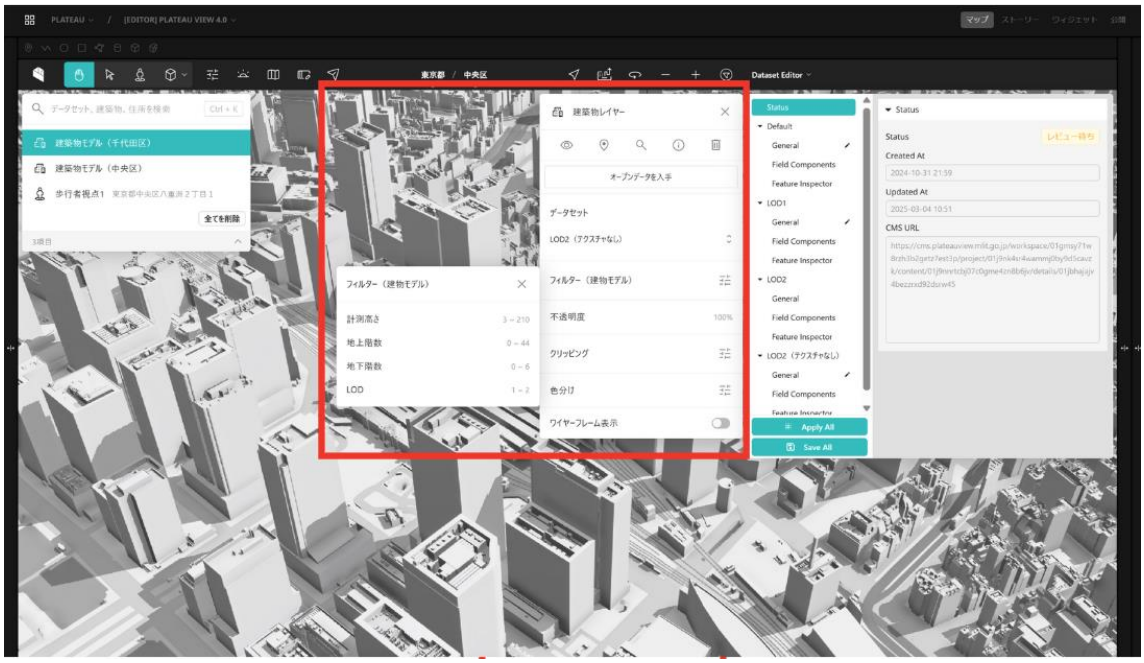
ヒエラルキーウィンドウでは、地図上に追加するデータを選択することができる。 3つのタブから構成され、検索、都道府県一覧から選択、カテゴリーから選択することができる。 Editorでのデータ確認時には、都道府県一覧から選択することをお勧めする.

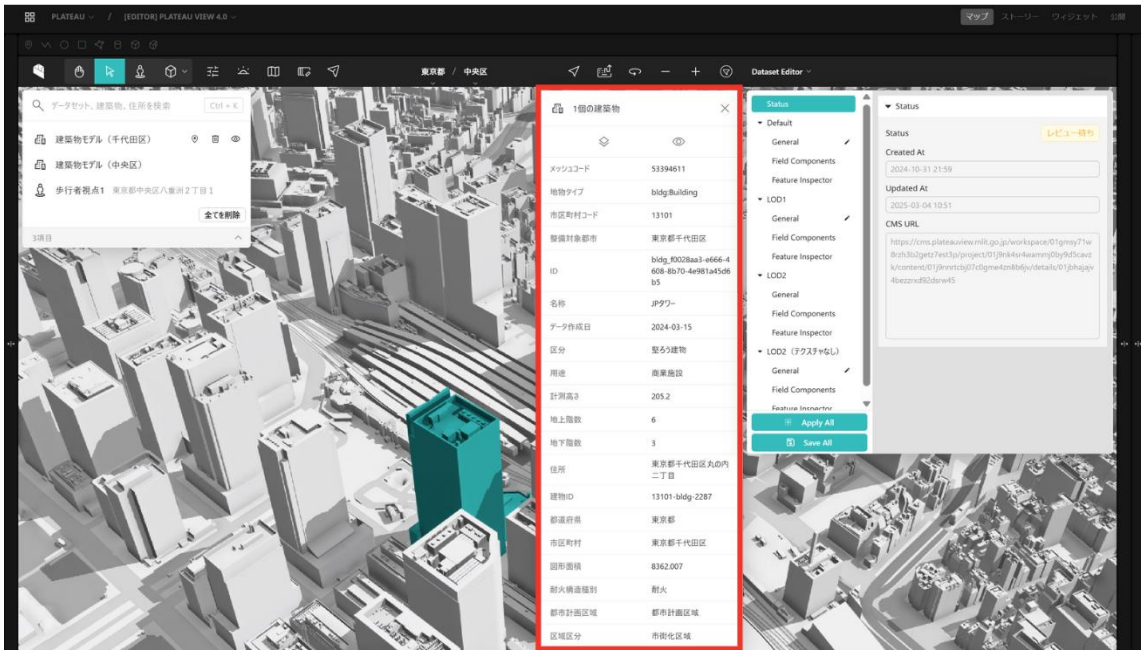
マウスのフォーカスを外すと 項目を選択できるタブが存在し、そこを押下すると、現在追加されているデーター一覧が表示される。

地図上における選択状態には2種類ある。レイヤーの選択と地物の選択である。 レイヤーを選択するには、データー一覧からレイヤーを押下する。 レイヤーの選択状態と地物の選択状態は、後述する。インスペクターの使い方に関わるので注意すること。

（3）インスペクター

上述したように、地図上における選択状態には、レイヤーと地物の選択状態が存在する。インスペクターは、レイヤー選択時には、レイヤーに関する情報を、地物の選択時には、地物の属性情報を表示する。レイヤーの選択状態のインスペクターからは、LODの色分け、クリッピング等の機能をすることができる。一方で、地物の選択状態では属性情報を閲覧することができる。データが正しいか確認するときには、この地物の情報をよく確認すること。



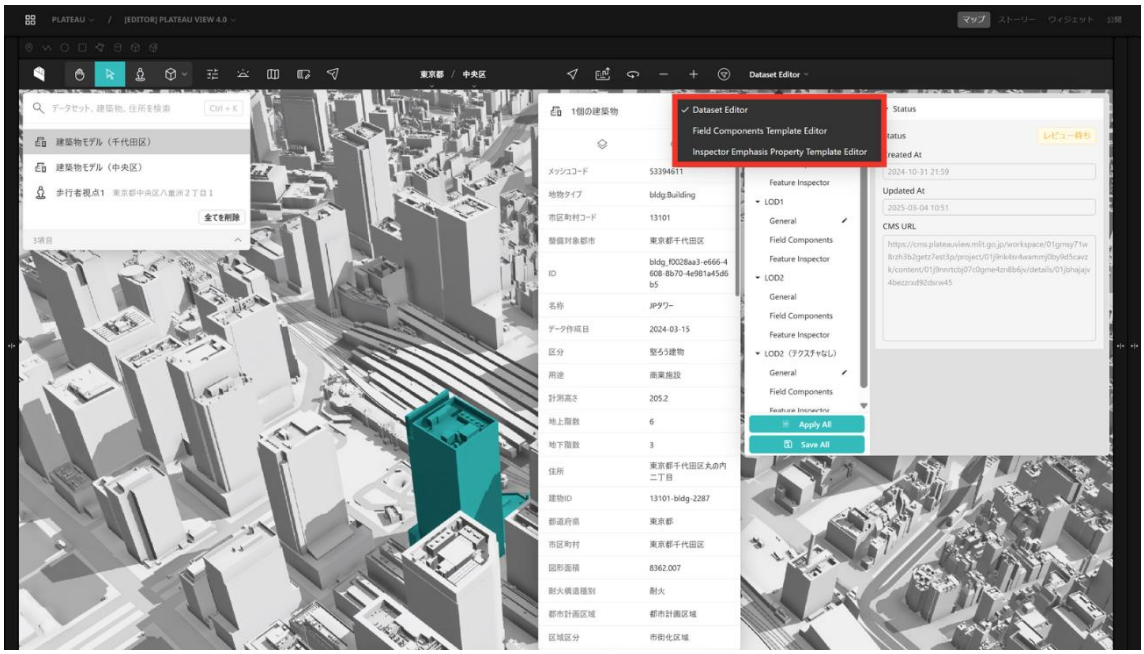


（4）データ編集パネル

コンポーネント設定パネルは、主に3つの設定ものが存在し、それぞれの設定の切り替えは画面スクショのタブから変更可能である。

1. コンポーネントEditor
2. テンプレートEditor
3. インスペクターEditor

以下にそれぞれの詳細を説明する。



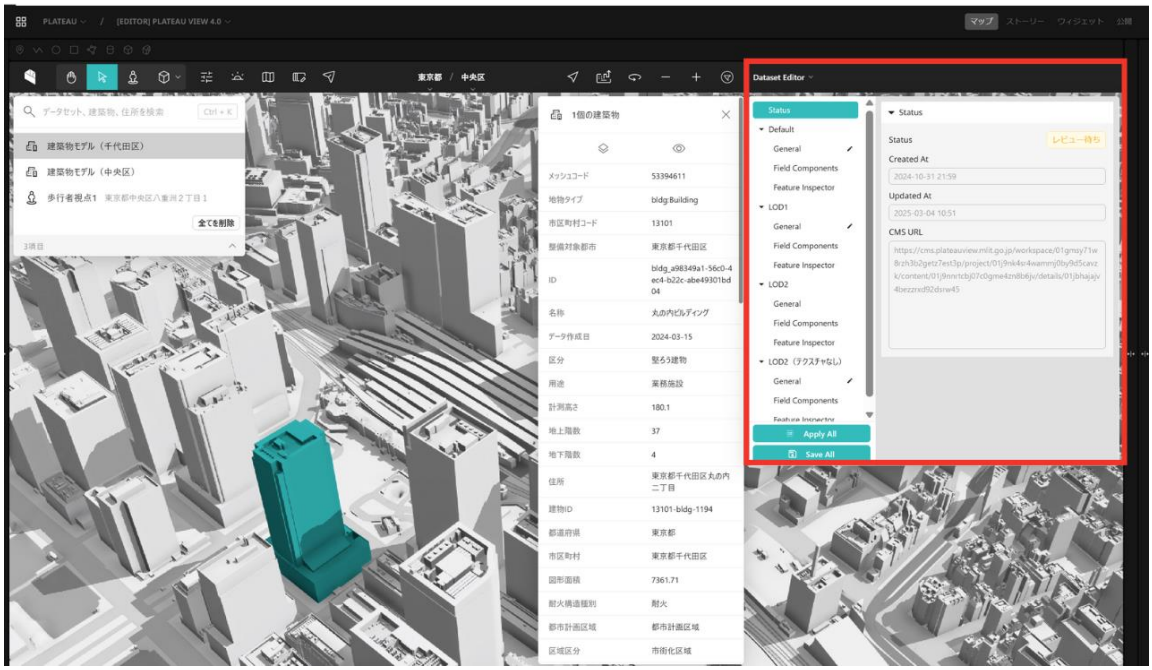
1.　コンポーネントEditor

Editorでは、データカタログから追加可能なそれぞれのデータセットに対し、UI操作によって凡例の表示や地図上のデータのスタイルを細かく設定することができる。それを可能にしているのが「コンポーネント機能」と「テンプレート機能」である。

コンポーネントとは？

コンポーネントとは、データセットに対し「凡例の表示」「スタイルの変更」など、Viewerにおける追加の機能を付加する仕組みである。データカタログで追加可能なそれぞれのデータセットに対し、複数のコンポーネントをひも付けることができる。
それぞれのコンポーネントは設定項目を持ち、VIEWではその設定項目を編集することはできないが、EditorではコンポーネントEditorで編集することができる。

　コンポーネントEditorは 最も多く使われる機能の1つであり、 凡例の表示は、地物の色分けの設定を行う。 設定方法は多岐にわたるため、本節では概要にとどめ、詳細な設定方法に関しては、2.2.2.3.コンポーネントの設定方法で解説をする。

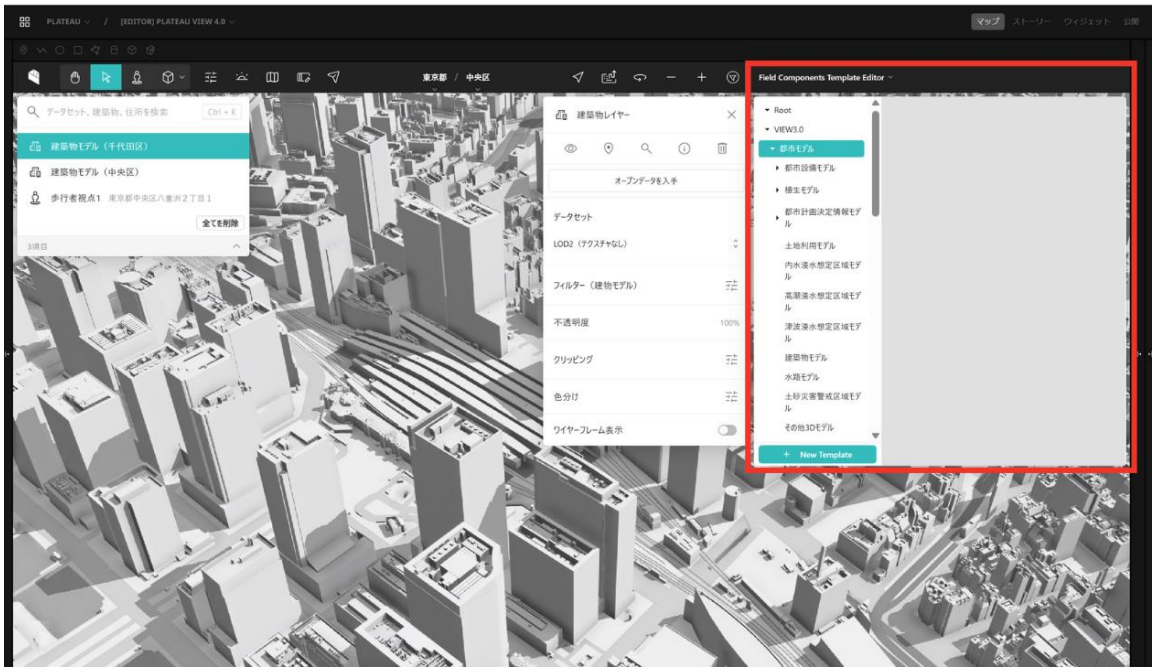


2.　テンプレートEditor

テンプレートEditorは、 利用可能なテンプレートの管理を行う場所である。

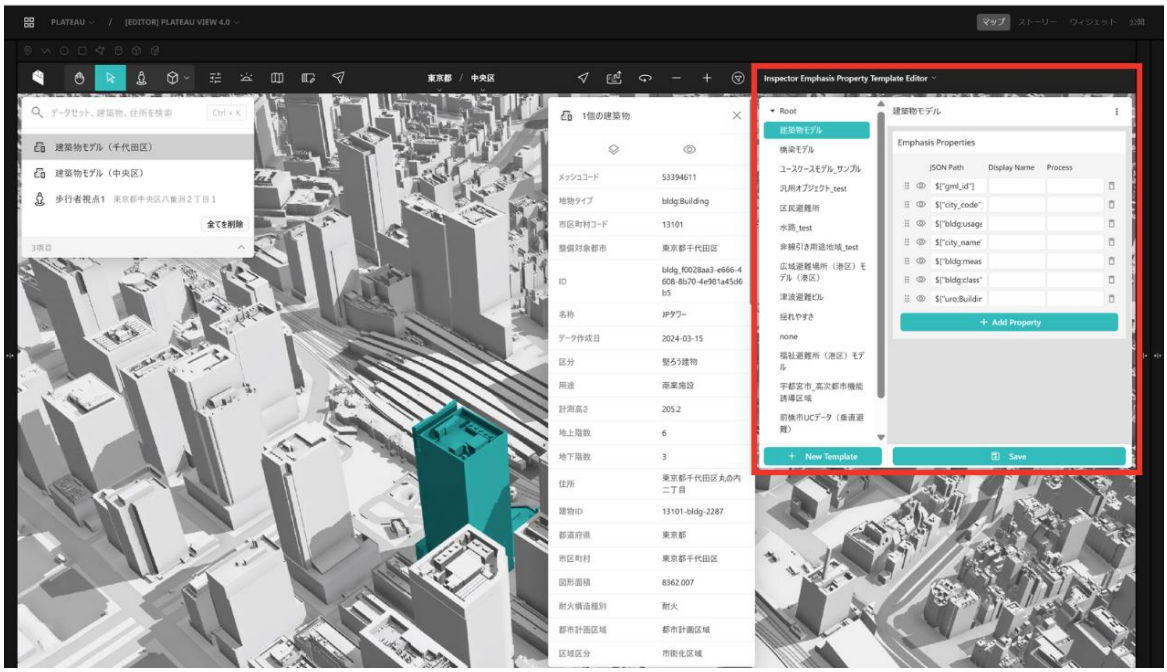
テンプレートとは？

　テンプレートとは、複数のデータセットに対し同じ設定のコンポーネントを一括で登録するための機能。データの種類に応じたテンプレートを作成することができ、データカタログからデータセットを追加すると、自動的にテンプレートで設定されたコンポーネントが反映される。
コンポーネントは複数の種類を同時に組み合わせることができ、どのような効果が発生するかはコンポーネントごとに異なる。例えばインスペクター上に凡例表示するための「凡例コンポーネント」は、地図上のデータのスタイルには何も影響を与えない。それに対し「ポイント>色コンポーネント」を使うと、サイドバーには特にUIは表示されないが、データの地図上での見た目を変更することができる。こうした異なる機能を持つコンポーネントを組み合わせることで、データセットに対しさまざまな表示のカスタマイズを行える仕組みとなっている



3. インスペクターEditor

インスペクターEditorとは、地物の属性表示方法をカスタマイズするための機能である。 データ整備事業者にとっては、利用機会が少ないが、 特定の属性を英名ではなく、日本語名で表示したい。また、特定の属性を表示にしたい等のケースにおいて利用できる主にユースケースデータでの利用を想定している。



2.2.2.3 コンポーネント設定方法

（1）コンポーネント設定の全体像

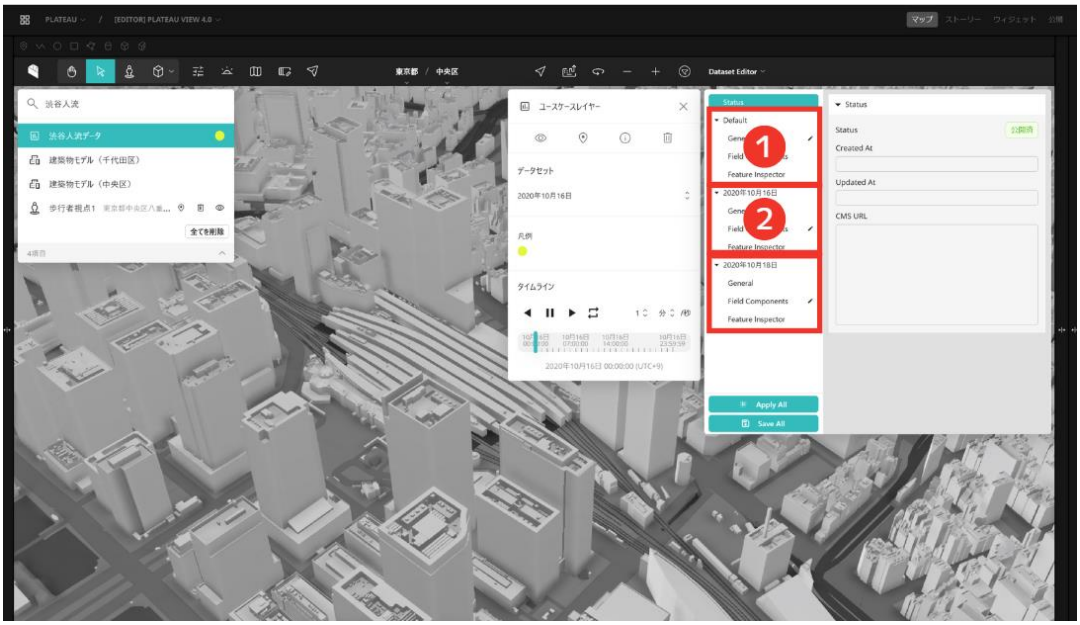
コンポーネントとは、データセットに対し「凡例の表示」「スタイルの変更」など、Viewerにおける追加の機能を付加する仕組みである。データカタログで追加可能なそれぞれのデータセットに対し、複数のコンポーネントをひも付けることができる。
それぞれのコンポーネントは設定項目を持ち、Viewerではその設定項目を編集することはできないが、EditorではコンポーネントEditorで編集することができる。

（2）コンポーネント設定をするデータ

コンポーネントは、各データ（データカタログに表示される単位）ごとに、それぞれ異なるコンポーネントを設定する。コンポーネントを設定する箇所は、大きく2つ存在する。

1. デフォルト
2. 個別データ

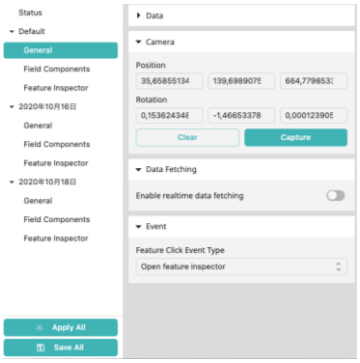
データカタログに表示されるデータは単一のファイルから構成されるデータもあれば、CMS上で複数のファイルを登録することで、複数ファイルから構成されるデータも存在する。（例：LODの切り替えなど）
原則としては、**「1.デフォルト」**でコンポーネント設定を行う。（多くのケースで「2.個別データ」への設定は不要）「1.デフォルト」で設定したコンポーネントの設定は、そのデータを構成する複数ファイル全てに適用される。一部、データを切り替えた際に適用するコンポーネントを変えたい場合のみ、対象のデータに対してコンポーネントを設定する。

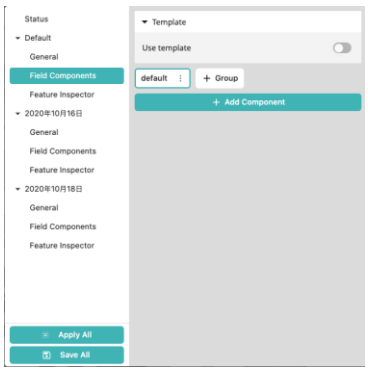


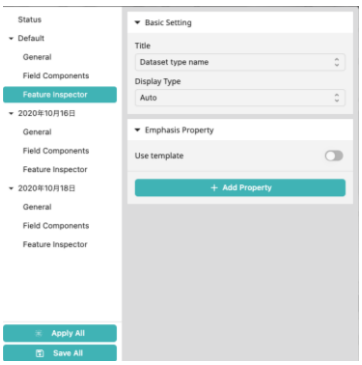
（3）コンポーネント設定の種類

さらにコンポーネントの設定には、３種類存在する。

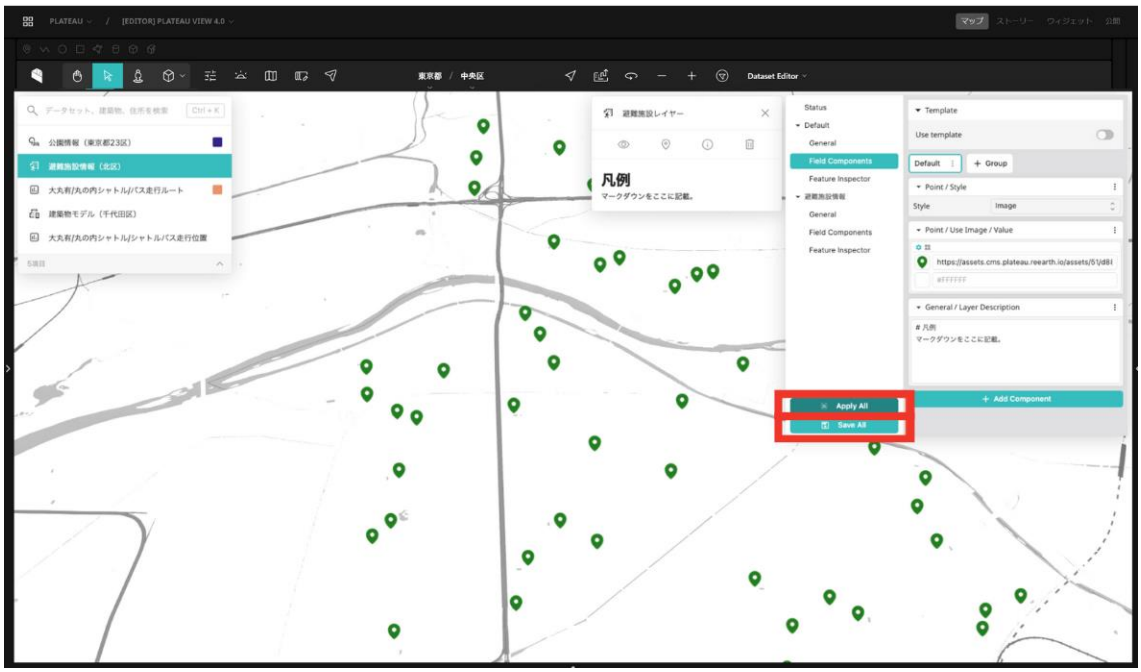
1. 一般設定: カメラの設定や、CMSから配信されるデータのJSON表記、地物押下時のイベントの設定などを行う。
2. フィールドコンポーネント: 地物の色分けや、凡例等の設定を行う。
3. 地物インスペクター: 地物の属性表示の設定を行う。







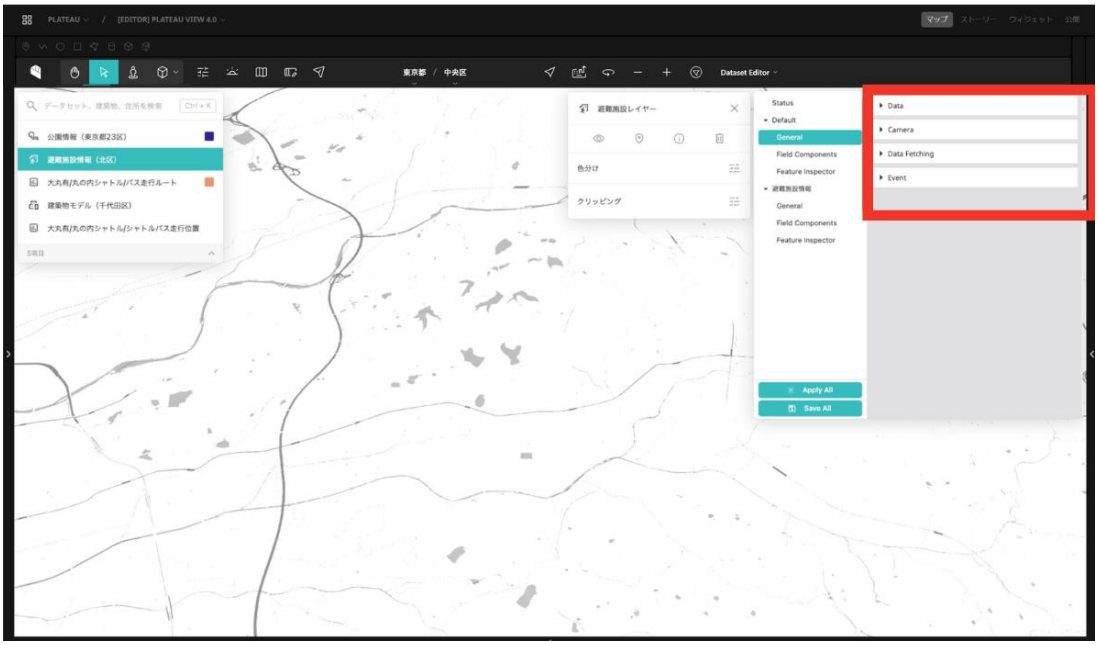
それぞれ設定が完了したら、「Apply All」を押下して設定を地図上の地物へ反映、「Save All」を押下して設定を保存する。（自動保存ではないため必ず保存を押すこと）



1.　一般設定

一般設定は、以下の4設定から構成される。

- データ
- カメラ設定
- データ取得設定
- イベント設定

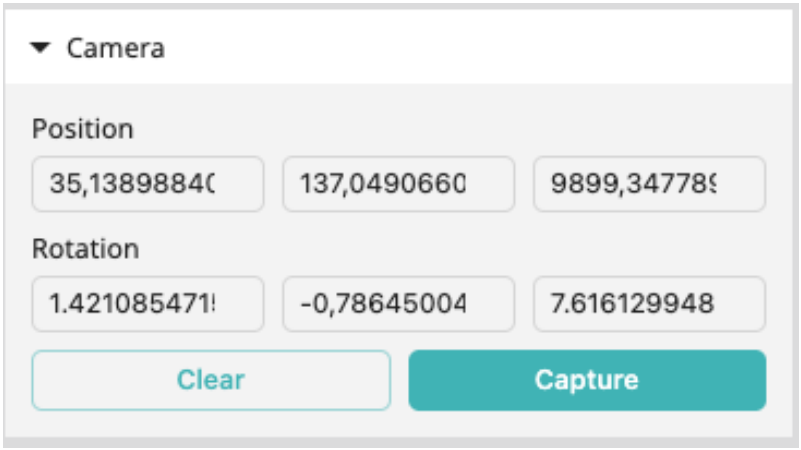


データ

CMSから配信されているデータの生データを確認することができる。主にデバッグ向け。

カメラ設定

「カメラ」ボタンで移動するカメラの位置を設定することができる。このコンポーネントがない場合は自動的にカメラ位置が計算される（MVTやWMSなどカメラ位置の自動計算が行われないデータフォーマットもある）。



データ取得設定

リアルタイムデータ取得を有効にする。GTFS Realtimeなどリアルタイムなデータ取得が必要な場合のみ有効にする。

イベント設定

地物押下時のイベントを設定する。

- Open feature inspector: 通常とおり、地物選択時に地物の属性情報表示をインスペクターから行う。
- Open new tab: 地物選択時、外部サイトへリンクしたい際に利用する。（利用頻度はあまり高くない）

2. フィールドコンポーネントの種類

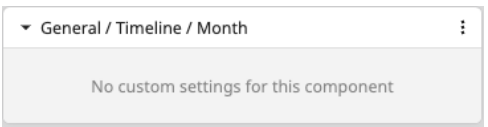
イベント設定

地物押下時のイベントを設定する。

- Open feature inspector: 通常とおり、地物選択時に地物の属性情報表示をインスペクターから行う。
- Open new tab: 地物選択時、外部サイトへリンクしたい際に利用する。（利用頻度はあまり高くない）

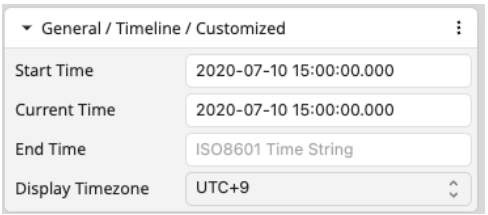
General > Timeline > Month

時系列データ表示用のタイムラインを表示する。1ヶ月前を開始日時、現在を終了日時として自動的に設定する。



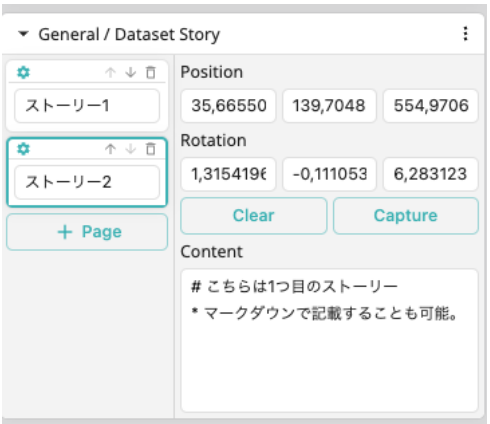
General > Timeline > Customised

時系列データ表示用のタイムラインを表示する。開始日時、現在日時、終了日時をISO8601形式で入力することで、再生バーの表示を行うことができる。



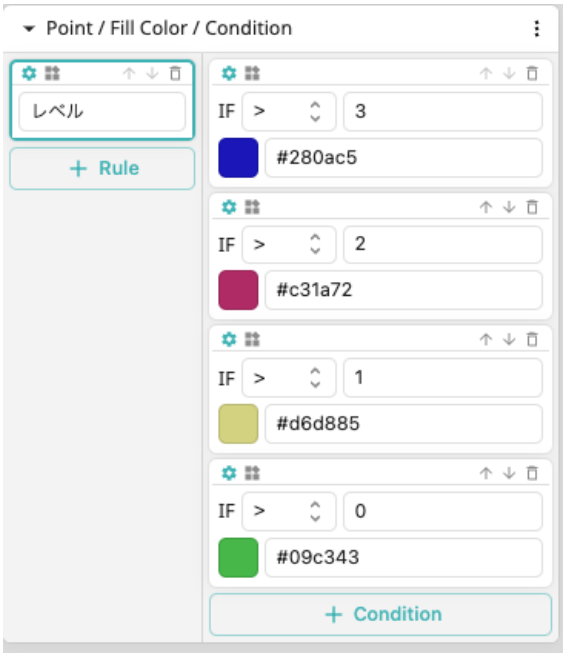
General > LinkButton
リンクボタンを表示する。タイトルとURLを設定することで、指定したリンクへ飛ぶリンクボタンを表示することができる。

General > Dataset Story
データセットにひも付くストーリーを作成可能。ストーリーは、複数のストーリーから構成され、それぞれのストーリーはカメラキャプチャとテキストを保持する。

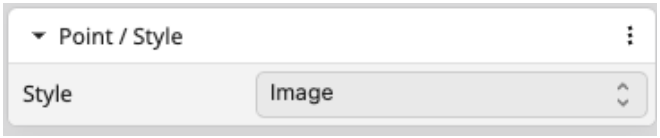


Point > Fill Color > Value
ポイント形式のデータにおいて、地物の色をカラーコードで設定する。（Line、Polygon形式のデータも同様のため省略）

Point > Fill Color > Condition
ポイント形式のデータにおいて、地物の特定の属性の値によって色を設定する。（Line、Polygon形式のデータも同様のため省略）

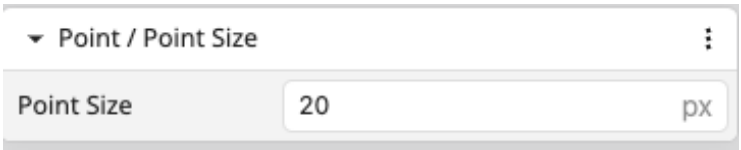


Point > StyleValue
ポイント形式のデータにおいて、地物の表示をポイントにするか画像にするかを選択する。データの表示においては必ず設定が必要。

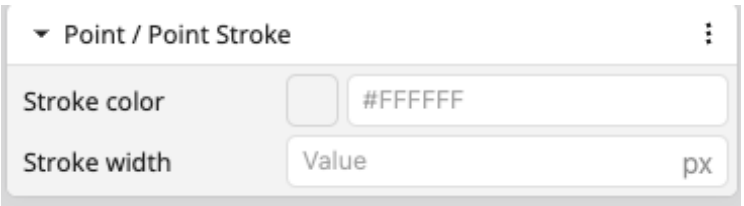


Point > Use Image > Value
ポイント形式のデータにおいて、地物表示に利用する画像のURLを指定する。

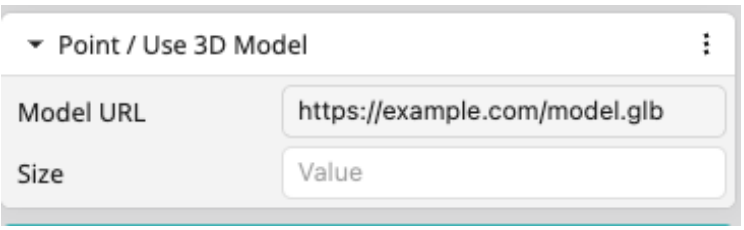
Point > Point Size
ポイント形式のデータにおいて、地物のサイズを指定する。



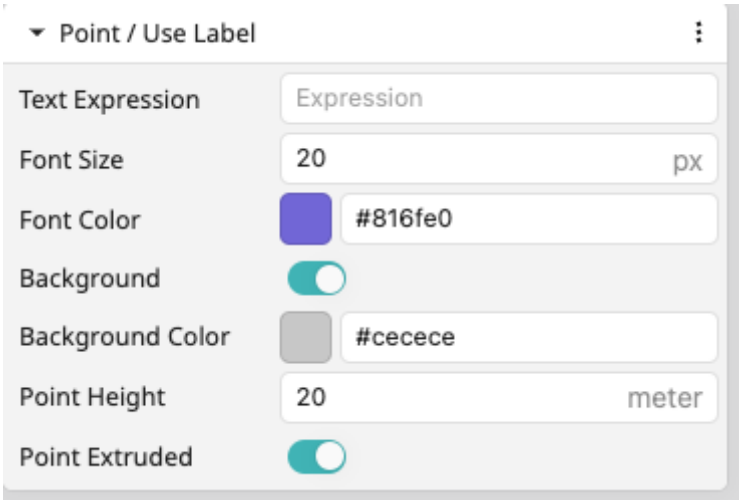
Point > Point Stroke
ポイント形式のデータにおいて、線の太さと色を設定する。



Point > Use Image > Value
ポイント形式のデータにおいて、3Dモデルを地物表示に利用する際に設定する。3Dモデル（glb形式）へのURLとサイズを設定する。

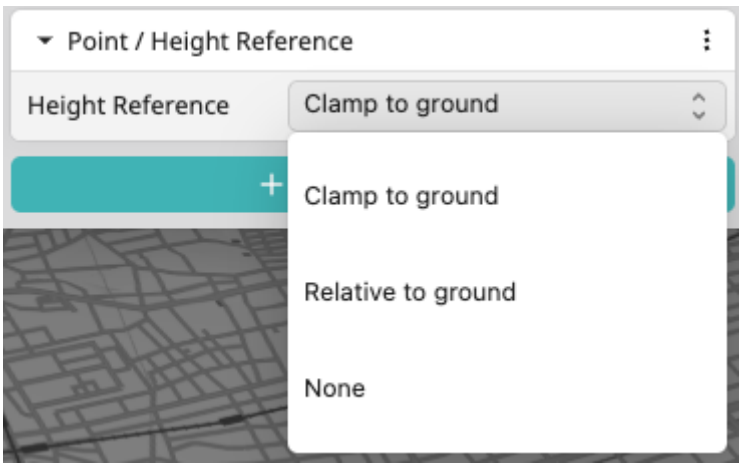


Point > Use Label
ポイント形式のデータにおいて、ラベル表示を行う。ラベル表示に利用する属性のキーと、その他フォント設定を行う。



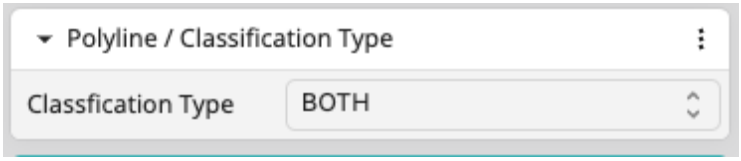
Point > Convert from CSV
CSV形式のデータをポイント形式で表示する際に利用する。緯度、経度、高さに利用する属性のキーを設定する。

Point > Height Reference
ポイント形式のデータにおいて、高さ位置設定を行う。
・Clamp to ground: 地表面にくっつくように高さが設定される。
・Relative to ground: 地表面から相対的な高さに設定される。
・None: 楕円体からの相対的な高さに設定される。

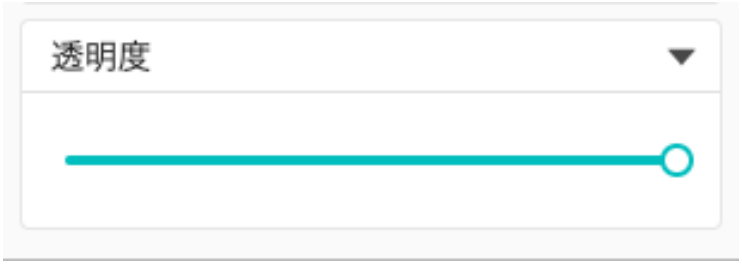


Line > Classification Type
ラインデータにおいて、地形や3Dモデルと位置的に重なるラインをどのようにドレープ表現するかを設定する。

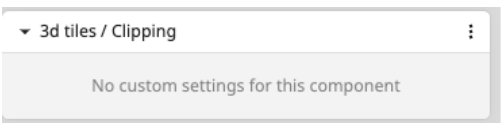
- Both: 3Dモデルと地形の両方に被るように表現する。
- Cesium 3D Tiles: 3D Tilesデータに被さるように表現する
- Terrain: 地形データに被さるように表現する。



3D Tiles > 透明度
3D Tilesの透明度を変更可能にする。

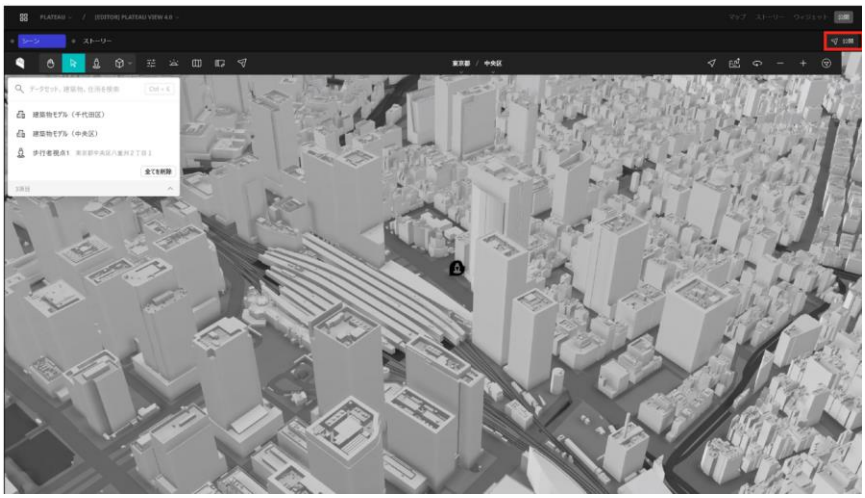


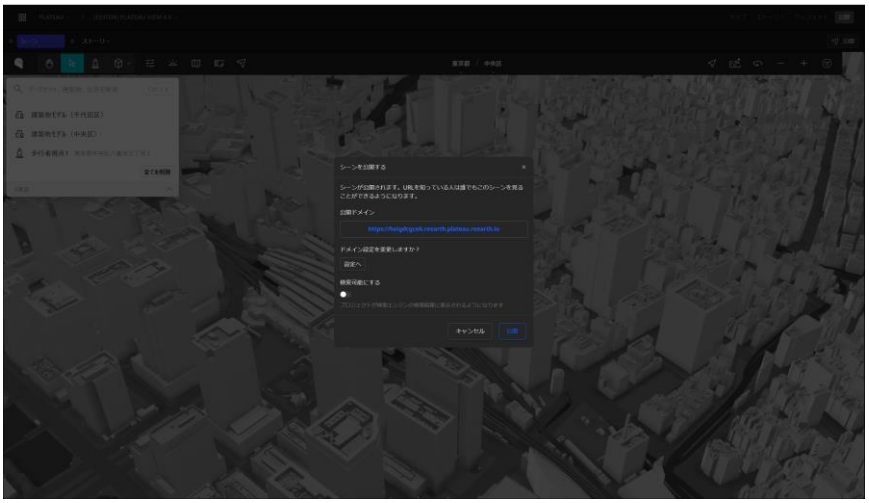
3D Tiles > Clipping
3D Tilesのクリッピング機能（マウスで操作可能な箱で建物などをくり抜いて表示する）を有効化する。

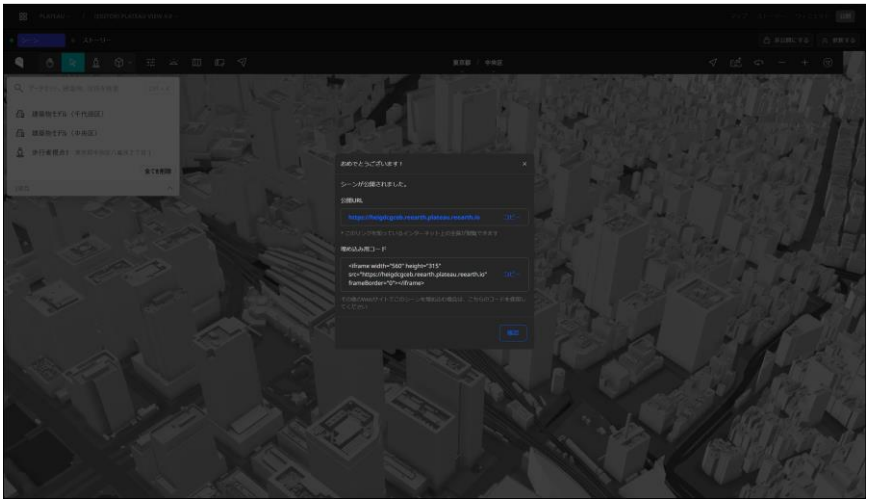


2.2.2.4 プロジェクトの公開

プロジェクト編集画面の右上の「公開」からプロジェクトを公開することができる。







2.2.2.5 データの可視化

PLATEAU VIEWで表示可能なデータフォーマットとそれらの表現方法のパターン、及びコンポーネントを概説する。

（1）主な対応データフォーマット

パフォーマンスの観点からWebで表示するのに適したフォーマットの使用を推奨する。ShapefileはWeb用に最適化されたフォーマットではないため、表示が遅いなどの理由で非推奨。なお、各GISファイルの属性やファイル内の文字コードは原則UTF-8で作成すること。

表　主な対応データフォーマット

フォーマット				説明
	単体	Zip 7Z	URL 指定	
GeoJSON	✓		✓	•JSON形式で記述されるGISファイルフォーマット。点、線、面のベクトルデータの表示に対応している。 ＜対応可能事項＞ •RFC7946で定義されたGeoJSON •ジオメトリのCRS：WGS84（EPSG:4326） ＜特記事項＞ 以下の形式については未対応である。 •GeoJSON 2008 •TopoJSON •CRSの指定*1 •標準仕様外のジオメトリ（Circleなど） *1：座標参照系（CRS:Coordinate Reference System）
CZML	✓		✓	•CesiumJS上でのデータ表現に対応したJSON形式のGISファイルフォーマット。 •CesiumJSの機能を使用してCZMLを表示するため、CZMLの仕様内でPLATEAU VIEW 4.0固有の制約はない。 ＜特記事項＞ •CZMLとその他関連するファイルをZipまたは7zファイルに圧縮・同梱することで、CZMLから相対パスで参照可能な別のデータセットを同梱することができる。この場合、CZMLはzipファイルのルート直下に置かれており、Zipファイル名と拡張子を除く部分が同じである必要がある。 （詳細は”CZMLをZip化する際のディレクトリ構成サンプル”を参照） •CZMLのdescription中では、相対パスや相対URLは使用できない。インフォボックス内に画像を表示させたい場合は、インターネット上で公開されている画像を絶対URL（httpまたはhttpsから始まるURL）で指定するか、CMSに画像だけを先にアップロードしてその画像の絶対URLを取得することで、使用することができる。
3D Tiles	✓		✓	•複数ファイルから構成されるデータフォーマットであるため、CMSにアップロードする場合は、それらをZipまたは7zファイルに圧縮する必要がある。 ＜特記事項＞ •圧縮する際には、ルート直下にtileset.jsonファイルを格納する必要がある。 ＜例外＞ •b3dmなど他のファイルはtileset.json内で相対パスが正しく定義されていれば、フォルダを挟んでも問題ない。

フォーマット				説明
	単体	Zip 7Z	URL 指定	
MVT		✓	✓	•拡張子は.mvt に対応する。 •複数ファイルから構成されるデータフォーマットなので、CMSにアップロードする場合は、それらをZipまたは7zファイルに圧縮してから必要がある。その場合、ルートから {z}/{x}/{y}.mvt のようにファイルを配置する。 <特記事項> •CMSでレイヤー名を指定しないと表示されない。レイヤー名はカンマ区切りで複数指定可能である。 •FMEでMVTへのデータ変換を行った際に出力されるmetadata.jsonに対応している。 •metadata.jsonを同梱する場合には、VIEWでのカメラボタンクリック時にカメラ位置をその内容に応じて自動的に移動することができる。 •metadata.jsonがルートに存在しない場合、カメラ移動は自動的に行われないため、予めEditorでカメラ位置を手動設定する必要がある。 •URL指定の場合、 {z}/{y}/{x}.mvt の指定がURL中にない場合は、それがURLの最後に自動的に付加されたものと同じ扱いになる。
Tiles		✓	✓	•複数ファイルから構成されるデータフォーマットであり、XYZ軸で分割された画像タイルである。CMSにZip形式でアップロードする場合は、ルートから {z}/{x}/{y}.png のようにファイルを配置する。 <特記事項> •URL指定の場合、 {z}/{y}/{x}.png の指定がURL中にない場合は、それらの文字列がURLの最後に自動的に付加される。
WMS (Web Map Service)			✓	<特記事項> •URLで指定する場合、レイヤー名の指定が必須となる。レイヤー名はカンマ区切りで複数指定可能。
TMS (Tile Map Service)		✓	✓	•複数ファイルから構成されるデータフォーマットであり、CMSにZip形式でアップロードする場合は、ルートから {z}/{x}/{y}.png のようにファイルを配置する。 <特記事項> •URLで指定する場合は、tilemapresource.xmlへのURLではなく、その親を指定する。 【誤】https://example.com/tms/tilemapresource.xml 【正】 https://example.com/tms •PNG画像のみ対応。拡張子は.png。tilemapresource.xmlが必須となる。
glTF（glb）	✓		✓	•拡張子は.gltf と.glb に対応。 <特記事項> •モデルの座標をCesium内部の座標系に合わせておく必要がある。事前に位置合わせを済ませたデータを使用すること。 •Web上で公開されているような通常のglTFでは座標系が異なるため正しく表示されない。
CSV	✓		✓ 単体のみ	<特記事項> •CSVファイル内のデータの1行目はヘッダとして扱われるため、各カラムの名前指定は必須となる。 •ジオメトリはポイントのみ対応する。ポイントの座標は、緯度・経度・高さでカラムを分けて数値で指定する。CMS上での登録時、これらのカラム名は自由だが、VIEWで正しく表示するにはEditorでのコンポーネント設定が必要であり、Editorでどのカラムを緯度・経度・高さとして扱うかをそれぞれ指定する。高さカラムは省略可能で、省略時は0として扱われる。 Editor側のカラム名は以下の通りである。 •ジオメトリ（緯度、経度、高さとして読み込むカラム）：at, lng, lon, height, alt, •スタイル（地図上のポイントのサイズと色）：pointSize, pointColor

（2）補遺

- 3D データの高さ方向の座標値について
- PLATEAU VIEWで表示する3Dデータを作成する場合、z座標は楕円体高（ジオイド高 + 標高）とする必要がある。
- PLATEAU VIEW 1.1からの変更点
- 1.1では内部システムにTerriaJSが採用されており、データカタログを記述したJSONを作成することで、凡例などのカスタマイズを可能にしていた。
- 2.0以降では独自のEditor及びVIEWを開発し、UI操作のみで凡例やスタイルのカスタマイズが可能になった。
- これに伴い、1.1で使用されていたツール「plateau-catalog-generator」は使用する必要がない。また、それを使用して生成したJSONはPLATEAU VIEW 2.0以降では使用できないことに注意。

2.2.3 PLATEAU VIEWの機能

PLATEAU VIEWでは主に以下の機能が利用可能である。詳しい使い方は以下の資料を参照されたい。
https://www.mlit.go.jp/plateau/learning/tpc02-1/#p2_5

データの追加と属性情報の表示

- ヒエラルキーウィンドウ
    - エリア/データセット検索
    - 追加したデータを表示・非表示
    - PLATEAU VIEWにデータを追加する
- インスペクター
    - 凡例の表示・絞り込み表示などの操作・地物の属性表示
    - 建物の高さや用途ごとに属性の色分け表示

地図の操作機能

- 基本的な地図操作
    - 地図の移動・ズーム
    - 地図の設定変更（ベースマップの切り替え、地図ラベルの表示）
- 視点とカメラの操作
    - 視点の変更
    - 歩行者視点の移動（Google Street Viewとの連携）
- データの活用と可視化
    - 統計データ・標高データのヒートマップ表示
    - 建築物モデルのワイヤーフレーム表示
    - 3D図形の作図
    - タイムラインによる時系列データの表示
    - ストーリーテリングの表示と編集
    - 空間IDから属性情報を取得 ※ View4.0より追加
    - 地域メッシュからCityGMLをダウンロード ※ View4.0より追加
- 環境設定
    - グラフィック品質設定
    - 太陽光設定
- その他便利な機能
    - コンパス機能
    - 現在地の表示
    - 建物のクリップ機能
    - URL共有機能
    - 任意の地理空間情報を追加する（Myデータ）
    - ご意見・ご要望

2.3　PLATEAU Flow

2.3.1 PLATEAU Flowとは

PLATEAU Flowとは、GISデータの変換や演算処理を行うためのシステムである。Web上で、コーディング不要で視覚的にデータ変換ワークフローを構築・実行できるという特徴を持つ。

特に、CMSに登録されたCityGML等のファイルを入力し、品質検査や3D TilesやMVTといったフォーマットへのデータ変換を行うために使われる。

（1）使用ソフトウェア・サービス

上記システムを構築するために、オープンソースソフトウェア（OSS）と、有償のクラウドサービスを組み合わせて利用している。クラウドサービスの詳細については次項の各コンポーネントの説明で述べる。

項目	項目	説明
Auth0	有償クラウドサービス	Auth0社が提供するクラウドサービス。アカウントの管理・認証・認可を行うIDプロバイダを提供する。Auth0を使用することで、開発者はアプリケーションに安全で使いやすく信頼性の高い認証・認可機能を組み込むことができる。PLATEAU FlowもAuth0を使用して認証・認可機能を実現している。
Google Cloud	有償クラウドサービス	Google社が提供するクラウドコンピューティングプラットフォーム。PLATEAU Flowを動作させるためのサーバーや、ファイルを保存するためのサーバーをGoogle Cloud上に構築している。
MongoDB Atlas	有償クラウドサービス	MongoDB社が提供するクラウドサービス。保守運用が自動化されたマネージドなMongoDBを提供している。MongoDBとは、ドキュメント指向のNoSQLデータベースで、データの柔軟性と拡張性が特徴。PLATEAU CMSはデータベースとしてMongoDBを使用している。

PLATEAU Flowはその他さまざまな技術を組み合わせて構築されている。

項目	説明
Go	プログラミング言語の1つで、Google社によって開発された。構文がシンプルであり、かつ処理が高速な言語であり、主にバックエンド開発に用いられる。PLATEAU FlowのAPIサーバー実装に利用されている。
TypeScript	プログラミング言語の1つで、Microsoftによって開発された。JavaScriptに静的型付けを加えたスーパーセットであり、大規模システムの開発に用いられる。PLATEAU Flowではフロントエンド向け開発に利用されている。
React	Meta社によって開発されたUI構築のためのJavaScriptライブラリ。特に大規模かつ複雑なUI実装において利用される。PLATEAU Flowではフロントエンド向け開発に利用されている。
GraphQL	API向けに作られたクエリ言語及びランタイムを指す。WebAPIの開発において、RESTなどの方式と比較して、より柔軟かつ効率的なAPIの提供を可能にする。PLATEAU Flowではバックエンドとフロントエンド間の通信に利用されている。

項目	説明
Rust	安全性とパフォーマンスに優れたプログラミング言語であり、メモリ管理の安全性を確保しながら高速な処理が可能。PLATEAU Flow のデータ処理部分で用いており、大規模な地理空間データの効率的な処理と管理を実現している。
Yjs	リアルタイム共同編集を実現するための JavaScript 向けの CRDT（Conflict-Free Replicated Data Type）ライブラリ。複数のユーザーがWeb上でプロジェクトを編集する際にスムーズな編集体験を提供している。
WebSocket	クライアントとサーバー間で双方向通信を行うためのプロトコル。PLATEAU FlowではYjsと組み合わせてリアルタイムな共同編集機能を実現している。
Terraform	HashiCorp社によって開発された、nfrastructure-as-codeを実現するためのソフトウェアであり、サーバー構成をコードとして宣言的に管理をできるようにする。

（2）PLATEAU Flowの主な機能

PLATEAU Flowでは主に以下の機能が利用可能である。詳しい使い方は、2.3.2を参照されたい。

- ワークスペースの作成・ユーザーの招待
- プロジェクトの作成
- ワークフローの編集
- ワークフローのデプロイ
- ワークフローの実行

2.3.2 ワークフロー構築・実行方法

（1）ワークフローの概要

ワークフローとは

ワークフローとは、一連の処理を視覚的に表し、データの流れや処理の手順を管理する仕組みのこと。

簡単なワークフローの例

ワークフローは、主にリーダー、トランスフォーマー、ライターの3つのアクションで構成される。以下の図は、ファイルのデータを読み込み、整理・変換し、最終的にGeoJSON形式でファイルを保存する一連の流れの例である。

1. **データの読み込み（FileReader） – リーダーの役割**
   リーダーは、外部ファイルからデータを取得し、後続の処理が行える状態に整える。この工程では、データをそのままシステムに取り込み、適切なフォーマットで処理の準備を行う。
2. **データの集約・整理（AttributeAggregator） – トランスフォーマーの役割**
   トランスフォーマーは、データを受け取り、必要な処理や変換を行う。今回は、特定の属性を基準にして、データ内の特定の属性（値）を基準にして、似た特徴を持つデータ（フィーチャー）をグループ化することで、整理・集約を行う。
3. **データの保存（GeoJsonWriter） – ライターの役割**
   ライターは、処理済みのデータを最終的な形式（GeoJSONなど）に変換し、ファイルに書き込みを行う。この工程により、データが適切なフォーマットで保存され、活用できる状態になる。

このように、リーダー（データの読み込み）、トランスフォーマー（データの処理・変換）、
ライター（データに書き込み）という3つの役割を通じて、効率的なデータ処理が実現される。

図　ワークフローイメージ

ワークフローの構築から実行までの主な手順

ワークフローの主な構築手順は以下の4つに分けられる。

1. プロジェクトの作成
    - ワークフローを管理するためのプロジェクトを作成する。
2. ワークフローの設計・構築（ワークフローエディター上で行う）
    - ワークフローの流れを設計し、必要なアクション（ノード）を追加する。
    - ノード同士を接続し、データの流れを設定する
3. ワークフローのテスト（動作確認）
    - ワークフローエディター上でワークフローを実行し、エラーがないか確認する。
4. ワークフローのデプロイ（本番環境で動作可能にする）
    - 構築したワークフローをシステムに保存し、運用可能な状態にする。
    - デプロイ済みワークフローを実行し、処理が正しく動作するか確認する。

2.3.2では、ワークフローの機能説明、具体的な操作方法を説明する。簡単なワークフローの例は2.3.4に記載しているため、あわせて参照されたい。

（2）PLATEAU Flowのプロジェクト作成

プロジェクトとは

プロジェクトは、データ処理を行うワークフローをまとめて管理するコンテナのこと。各プロジェクトには、データの処理手順、設定、およびデプロイ情報が含まれる。

プロジェクト作成手順

ワークフローの主な構築手順は以下の4つに分けられる。

1. PLATEAU Flowにログインする。
    - ログイン後、ダッシュボード画面が表示される。

2. ダッシュボード左上の 「プロジェクト」 をクリックし、プロジェクト一覧を開く。
3. ダッシュボード右上の 「+ 新規プロジェクト」 ボタンをクリックする。
4. 表示されたダイアログにて、プロジェクト情報を入力する。
    - プロジェクト名: 任意のプロジェクト名を入力する。
    - プロジェクトの説明（オプション）: プロジェクトの目的や概要を記入する。
5. 「作成」ボタン をクリックすると、プロジェクトが作成される。
6. 作成後、プロジェクト一覧に表示され、プロジェクトをクリックすると編集画面が開く。

（3）ワークフローエディターの概要

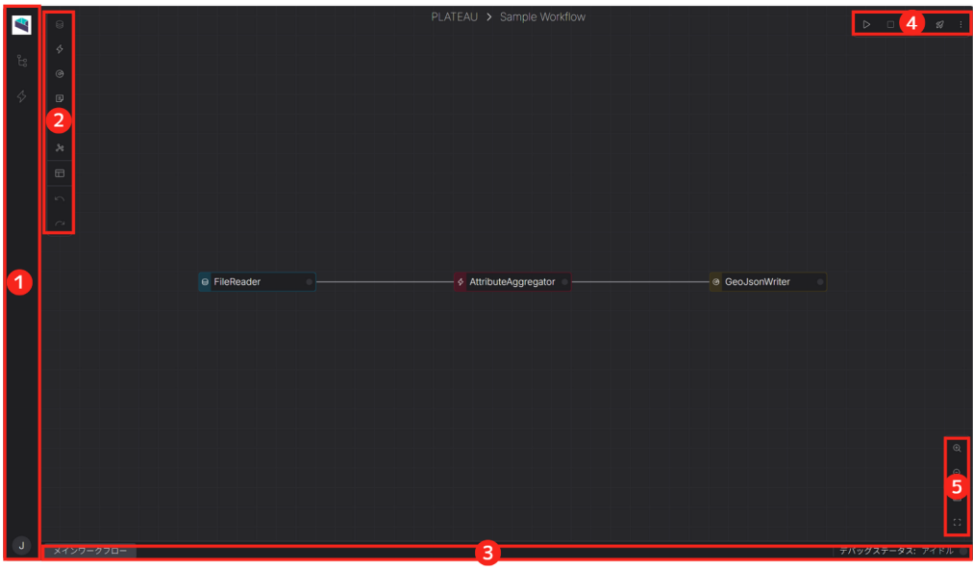
ワークフローエディターとは

ワークフローエディターでは、ワークフローを視覚的に構築・設定することができる。

画面構成

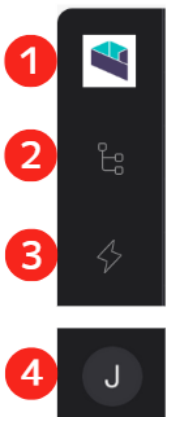
1. 左サイドバー:ワークフローの管理や設定変更を行うための主要な機能にアクセスできる。
2. キャンバスツールバー:ワークフローの構築に必要なノードをを追加・管理できるツール群。
3. 下部パネル:ワークフローの管理や、実行状況の確認ができるパネル。
4. アクションパネル:ワークフローの実行や管理に関する操作を行うためのパネル。
5. キャンバスコントロール:ワークフローの全体表示を調整できるエリア。

図　エディタの画面構成



機能説明

- 左サイドバー



機能名称	説明
1.PLATEAU Flow アイコン	ダッシュボード（トップ画面）に戻る。
2.キャンバスファイルマネージャー	ファイルをツリー形式で整理し、ワークフローを管理できる。
3.トランスフォーマーパネル	利用可能なアクションリスト を表示する。
4.ユーザー設定	アカウント設定や環境設定 を開く。

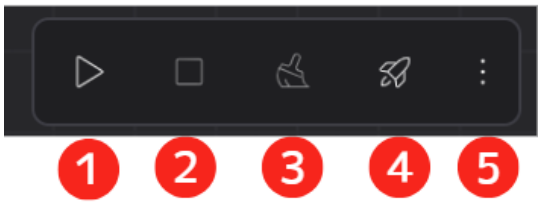
- キャンバスツールバー

機能名称	説明
1.リーダーノード	ワークフローで使用する入力データを読み込む（プロジェクトごとに1つのみ）。
2.トランスフォーマーノード	データを受け取り何らかの処理を行い結果をデータとして出力する。
3.ライターノード	ファイルなどのデータに書き込む。
4.ノートノード	キャンバス上にメモを追加する。
5.バッチノード	複数のノードをグループ化する。
6.サブワークフローノード	再利用できる独立した小規模なミニワークフロ ーを構築する（InputRouterとOutputRouterが必要）。リーダーやライターノードはサブワークフロー内で使用不可。
7.自動レイアウト	キャンバス内のノードをDagretreeのアルゴリズムを使って自動的に水平（横並び）または垂直（縦並び）に並べる。
8.元に戻す/やり直し	直前のアクションを取り消しや復元を行う。

- 下部パネル

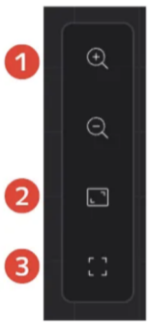
機能名称	説明
1.ワークフロータブ	ワークフローエディター内で、構築・管理しているワークフローを一覧表示し、切り替えや操作を行うためのタブ。メインとサブの2種類ある。
2.デバッグステータス	ワークフローの動作状況や不具合の有無を確認できる。

- アクションパネル



機能名称	説明
1.ワークフローのデバッグ実行の開始	ワークフローをデバッグ実行を開始する。
2.ワークフローのデバッグ実行の停止	ワークフローのデバッグ実行を停止する。
3.デバッグ実行と結果をクリア	デバッグ実行後に表示された結果（ジョブ/ノードステータス・ログ等）をリセットできる。
4.ワークフローのデプロイ	ワークフローをデプロイする。
5.⋮（その他のアクション）	プロジェクトの共有URLの発行、エクスポートやバージョン履歴を管理。

- キャンバスコントロール



機能名称	説明
1.ズームイン/ズームアウト	ワークフローの表示サイズを拡大/縮小する。
2.ビューモード内のすべてのノード	すべてのノードが画面内に表示されるよう、自動で中央へ移動する。
3.フルスクリーンに入る	エディタを全画面表示する。

（4）ワークフローの構築

ワークフローエディターでは、データの処理手順を設計し、ノードを追加・接続することでワークフローを構築する。ワークフローの構築は、まず処理の枠組みとなる「ノード」を追加し、そのノードに実行する処理である「アクション」を設定し、適切に接続することで完了する。

ノードとアクションの関係性

ワークフローにおいて「ノード」と「アクション」は異なる役割を持ち、相互に関係している。ただ、ワークフローの作成では、まず「ノード」を追加し、その中で適切な処理を実行する「アクション」を設定することが基本となる。

- ノード: ワークフロー上に配置される視覚的なブロックで、処理を実行するための枠組み（例：リーダーノード、トランスフォーマーノード）。
- アクション: ノード内で実行される具体的な処理（例：特定のデータを抽出する、データを書き出す）。

ノードの追加

ワークフローエディター内にデータの処理を行うノードを追加する。
ノードを追加する方法は、以下の2種類がある。

- **左サイドバーからノードを追加する方法**

1. 左サイドバー内の「トランスフォーマーパネル」 をクリックする。
2. 表示されたアクション一覧から、追加したいアクションを選択する。
    - アルファベット順で検索: リストがアルファベット順に並び替えられたリストから検索可能。
    - カテゴリ別で検索: アクションを機能ごとに分類し、用途に応じたグループから選択できる。（例: Attribute, Debug, File, Geometry, PLATEAU など）
    - 種類別で検索: アクションの処理の種類に基づいて絞り込み、特定の動作に適したアクションを探すことができる。（例: Transformer, Writer）

- **キャンバスツールバーから追加する**

1. キャンバスツールバーから追加したいノードを選択し、キャンバス上へドラッグ＆ドロップする。
2. アクションダイアログが開くので、追加したいアクションをダブルクリックして選択する。

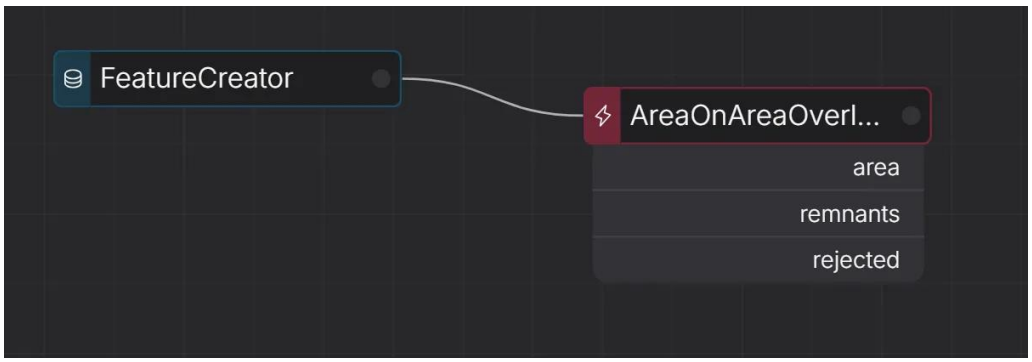
以下のキーボードショートカットを押すと、ノードの選択操作を省略し、指定のアクションダイアログを直接開いてアクションを追加できる。

- R: リーダー（Readers）を開く。
- T:トランスフォーマー（Transformers）を開く。
- W:ライター（Writers）を開く。

ノードの接続

ワークフロー内で複数のノードのアクションをつなげて、データの流れや処理の順序を定義する。
ノード同士を線でつなぐことで、データの流れを定義し、ワークフローを完成させる。

図　ノード同士の接続



接続方法

- ノードアクションの出力部分（右端）をクリックし、別のノードアクションの入力部分（左端）へドラッグ＆ドロップする。

接続の削除

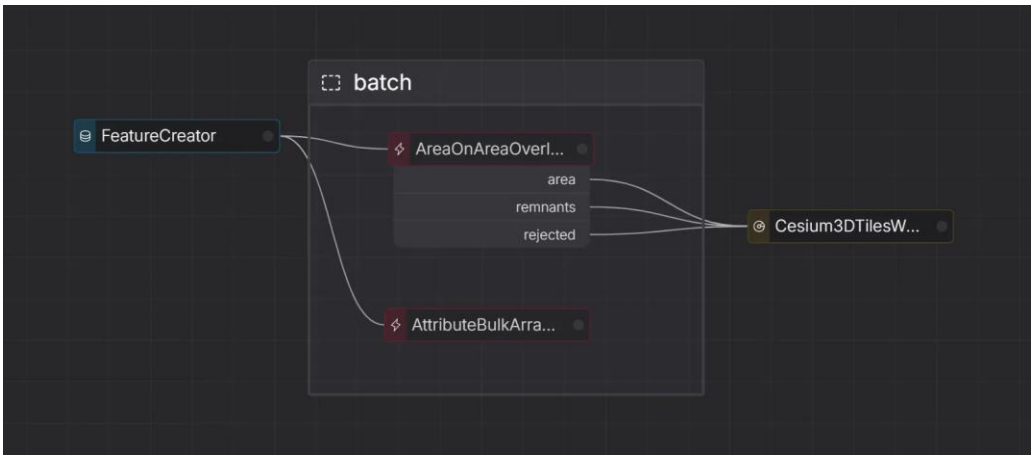
- 2つのノード間の接続を選択し Backspace キーを押す。

ノードのグループ化

バッチノードを使用すると、複数のノードをグループ化して整理できる。

- キャンバスツールバーからバッチノードを選択し、キャンバス上へドラッグ＆ドロップする。
- バッチノードを削除すると、その中に含まれるすべてのノードも一緒に削除される。

図　バッチノードを活用したノードのグループ化



ノードの編集

アクションの詳細設定をカスタマイズする。

- アクションをダブルクリックすると、パラメータエディタが開き、設定を行える。

ノードの削除

キャンバス上からノードを削除する。

- 削除したいノードを選択し、Backspaceキーを押す。
- 削除したノードが他のノードと接続されていた場合、接続も同時に削除される。

（5）ワークフローのデプロイの概要

ワークフローのデプロイとは

デプロイとは、構築したワークフローをシステムに登録し、実際に動作可能な状態にする作業である。デプロイ済みのワークフローを実行すると、データ変換や処理が実際に行われるようになる。

ワークフロー、デプロイ、デプロイ済みワークフローの関係

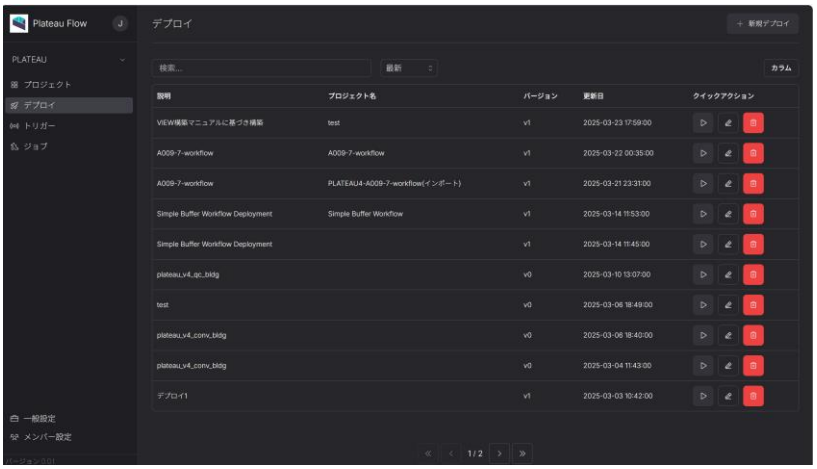
ワークフローのデプロイに関連する用語を以下のように定義する。

- ワークフロー
    - データの処理手順を定義したもの。
    - ワークフローエディターで構築し、各ステップでデータの変換や処理を行う。
    - 構築しただけでは実行できない。
- デプロイ
    - ワークフローをシステムに登録し、実行可能な状態にすること。
    - デプロイ後に初めてワークフローが動作できるようになる。
- デプロイ済みワークフロー
    - デプロイが完了し、実行可能になったワークフロー。
    - 管理画面から実行し、ジョブとして処理が進行する。

デプロイの管理画面

デプロイの管理は「デプロイ一覧ページ」または「個別デプロイページ」で行う。一度に複数のデプロイ済みワークフローを管理したい場合はデプロイ一覧ページから、特定のワークフローの実行履歴や詳細を確認しながら実行したい場合は個別デプロイページから実行するのが便利である。

- デプロイ一覧ページ:デプロイ済みワークフローを一覧で確認し、実行・編集・削除が可能である。



- 個別デプロイページ:デプロイ済みワークフローの詳細情報や実行履歴を確認でき、実行・編集・削除が行える。

(6)ワークフローのデプロイ手順

新規デプロイの作成

1. ワークフローエディターの画面右上にある ロケットのアイコン（デプロイボタン） をクリックする。
2. 表示されたダイアログボックスに、デプロイの説明を入力する。
3. 「更新」ボタン をクリックして、デプロイを開始する。
4. デプロイが成功すると、画面右下に 「デプロイメント作成完了」 と表示される。

デプロイの確認

デプロイが正常に完了したかを確認する。

1. 左サイドバー内のFlowアイコンをクリックし、ダッシュボードへ戻る。
2. ダッシュボード画面の左側メニューから「デプロイ」タブを開き、ワークフローが正常にデプロイされていることを確認する。

デプロイの更新

デプロイの更新を行うことで、既存のワークフローを新しいバージョンに変更できる。

1. 「デプロイ一覧ページ」から更新する場合
    - 更新対象のワークフローの欄にある「クイックアクション」メニューから 鉛筆のアイコン（デプロイメントを編集ボタン）をクリックする。
2. 「個別デプロイページ」から更新する場合
    - 画面右上にある 「デプロイメントを編集」ボタンをクリックする。
3. 表示される編集ダイアログで、以下の項目を変更できる。
    - 新しいワークフローファイルのアップロード

    注意:意図しない変更や不具合が生じる可能性があるため、ファイルによる更新は推奨されない。可能な限り使用を避けること。また、元々ファイルを使用せずに作成されたデプロイでは、このオプションは表示されない場合がある。
    - 説明の変更
4. 「デプロイメントを更新」ボタンをクリックし、変更を保存する。
    - 更新が完了すると、画面右下に 「デプロイメント更新完了」 の通知が表示される。

デプロイの削除

作成したデプロイを削除する。

1. 「デプロイ一覧ページ」から削除する場合
    - 削除対象のワークフローの欄にある「クイックアクション」メニューから ゴミ箱のアイコン(削除ボタン)をクリックする。
2. 「個別デプロイページ」から削除する場合
    - 画面右上にある「削除」ボタンをクリックする。
3. 確認プロンプトが表示されるので、削除を確定する。

注意:削除を実行すると、デプロイメントとそのデータがサーバーから完全に削除されます。

（7）デプロイ済みワークフローの実行方法

デプロイ済みワークフローの実行

デプロイが完了すると、作成したワークフローを実行できる。デプロイ済みワークフローを実行することで、デプロイされた処理がシステム上で実際に動作し、設定されたデータの変換や処理が順番に行われる。

1. 「デプロイ一覧ページ」から実行する場合
- 実行対象のワークフローの欄にある「クイックアクション」メニューから「▷」 のアイコン(実行ボタン)をクリックする。

2. 「個別デプロイページ」から実行する場合
- 画面右上にある「実行」ボタンをクリックする。

実行後の確認:

- デプロイ済みのワークフローを実行すると、その処理が「ジョブ」として自動生成され、ジョブ詳細ページに自動的に移動する。
- ジョブ詳細ページで、実行状況や出力結果を確認できる。

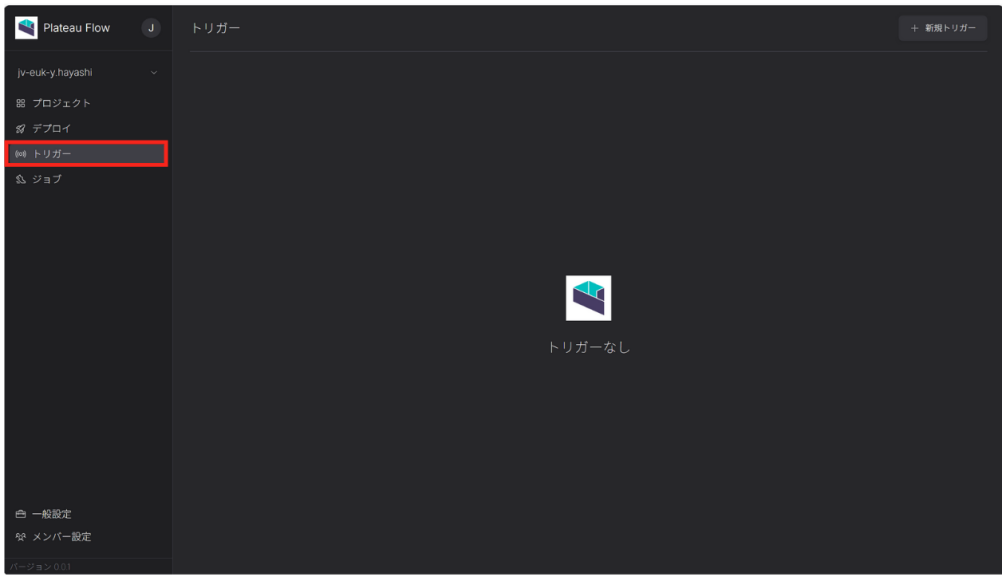
（8）ワークフローのトリガー

トリガーとは

トリガーとは、特定の条件が満たされたときや外部システムからの指示を受けたときに、自動でデプロイ済みワークフローを実行する仕組みである。これにより、手動での操作を省略し、作業の自動化や効率化が可能となる。 現在、主なトリガーの種類として 「APIドリブン」 がある。これは、外部のシステムやサービスがAPIを通じてデプロイを実行する方法であり、外部ツールと連携して自動化を進めることが可能である。

新規トリガーの作成

1. ダッシュボードから「トリガー」タブへ移動する。



2. 画面右上にある「新規トリガー」ボタンをクリックする。
3. 表示された作成ダイアログにトリガーの詳細情報を入力する。
    - 説明: トリガーの目的がわかるような説明を入力する。
    - デプロイメント: トリガーが発生した際に実行するデプロイ（ワークフロー）を指定する。
    - イベントソース: トリガーの発生条件を設定する。
        - APIドリブン:API経由でワークフローを自動実行する。主にPLATEAU CMSと連携する際に活用する。
        - タイムドリブン: スケジュールを設定し、指定した時間にワークフローを自動実行する。
4. 「新しいトリガーを追加」ボタンをクリックする。

PLATEAU CMSとの連携

トリガーを活用し、PLATEAU CMSとPLATEAU Flowを連携することで、CMS上でデータを管理しながら、自動的にワークフローを実行することが可能になる。

ただし、OSSから環境構築した場合、場合、この連携は自動的に再現されないため、PLATEAU CMS上にコンテンツ「Flowテスト用」を作成し、個別にトリガーを設定する必要がある。

連携手順

- 「Flowテスト用」コンテンツに新規アイテムを作成し、都市名や地物型など必要なデータを登録する。
- 「Flow連携」ボタンを操作し、データ変換を開始する。
- CMSのコメント機能を利用して、Flowの変換完了通知を確認する。

（9）ワークフローのジョブ

ジョブとは

ジョブとは、ワークフローが実行される際に作成される処理の実行単位であり、実行状況の追跡、ログの記録、出力結果の管理を行うもの。

ジョブの仕組み

- ワークフローを実行すると、システムがジョブを作成する。
- ジョブのステータスによって処理の進行状況がわかる。
- ログを確認すれば、エラーが発生したときの原因を特定できる。

ジョブの確認方法

ジョブの確認方法には、すべての実行済みジョブを一覧で確認できる「ジョブ一覧ページ」 と、各ジョブの詳細情報を確認できる「ジョブ詳細ページ」 の2つがある。各ページについて以下で詳しく説明する。

ジョブ一覧ページ

ジョブ一覧ページでは、すべての実行済みジョブが表示される。

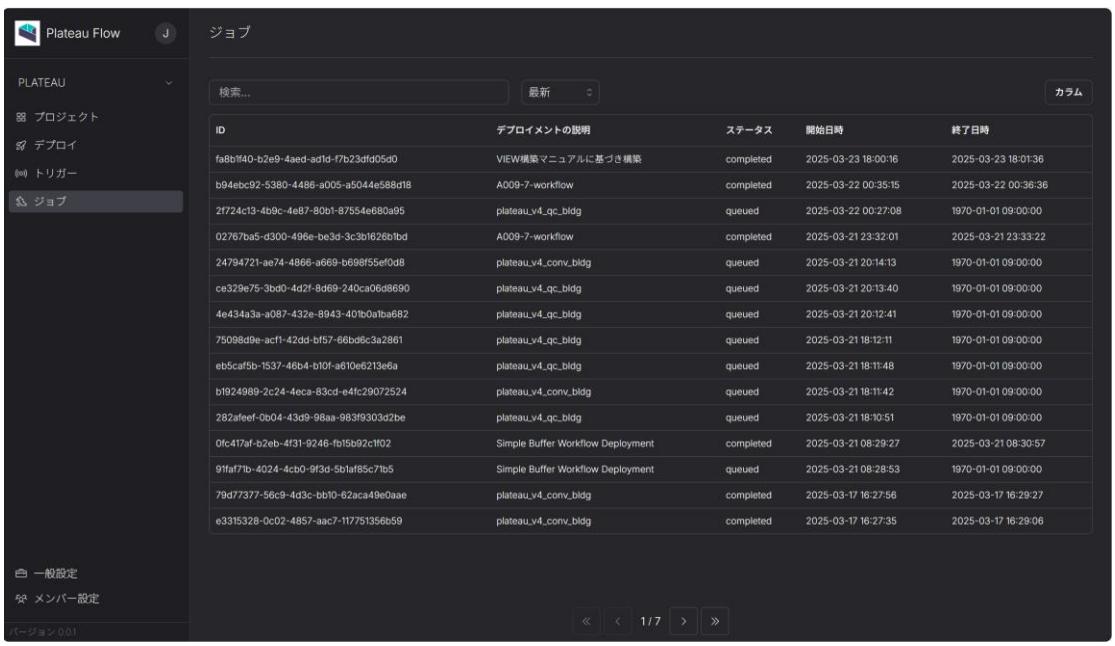
ジョブ一覧ページの表示方法

- ダッシュボードの左側にあるメニューから「ジョブ」をクリックする。

ジョブ一覧に表示される情報

項目	説明
ジョブID	ジョブの一意の識別子
プロジェクト名	ワークフローが属するプロジェクト名
ステータス	ジョブの現在の状態（詳細は次の項目参照）
開始時間	ジョブの実行が開始された時間
完了時間	ジョブが終了した時間

図　ジョブ一覧ページ



ジョブのステータス
ジョブのステータスは以下のいずれかになる。

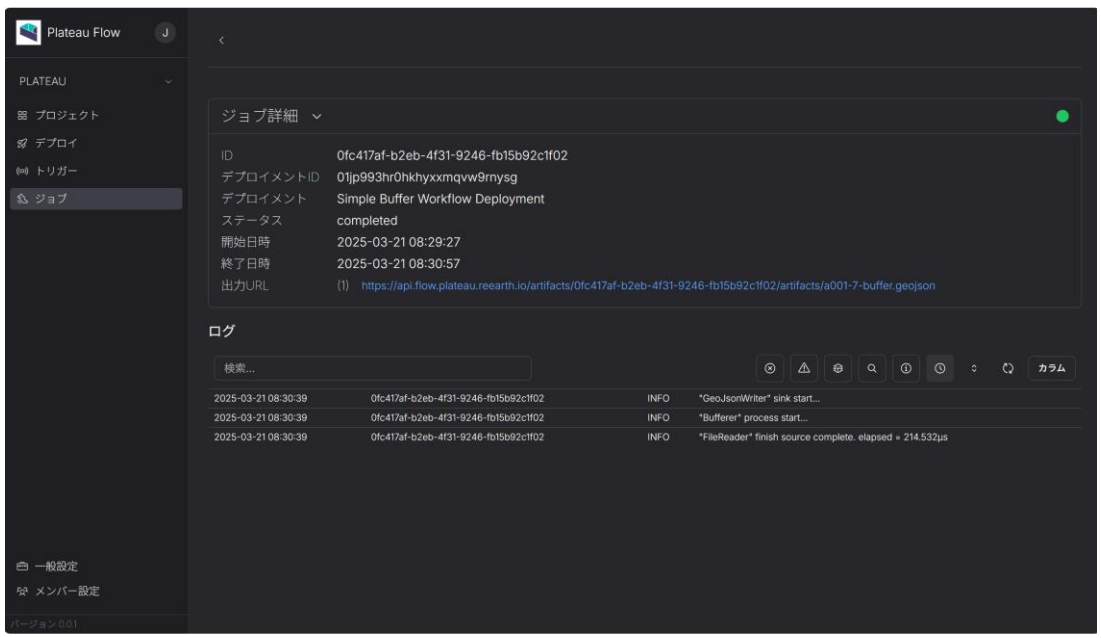
ステータス	説明
完了（Completed）	正常に実行されたジョブ
待機中（Queued）	実行待ちのジョブ
失敗（Failed）	エラーが発生したジョブ
実行中（Running）	現在進行中のジョブ
キャンセル済み（Cancelled）	手動で停止されたジョブ

ジョブ詳細ページ
ジョブの詳細ページでは、個々のジョブの詳細な情報を確認できる。
ジョブ詳細ページの表示方法
- ジョブ一覧ページで対象のジョブをクリックすると、詳細ページが表示される。

ジョブ詳細ページに表示される情報

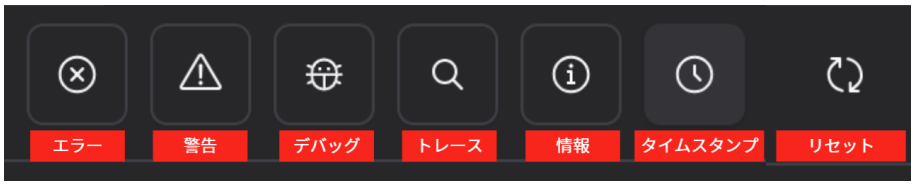
項目	説明
ID	ジョブの一意の識別子。特定の実行を追跡するのに使用。
デプロイメントID	このジョブをトリガーしたデプロイ済みワークフローのID。ワークフローの実行と関連付けられる。
ステータス	ジョブの現在の状態（完了・待機中・実行中・失敗・キャンセル済み）。
開始時刻	ジョブの実行が開始された時間。
完了時刻	ジョブが終了または停止された時間。
出力URL	ジョブによって生成された出力結果へのリンク。

図　ジョブ一詳細ページ



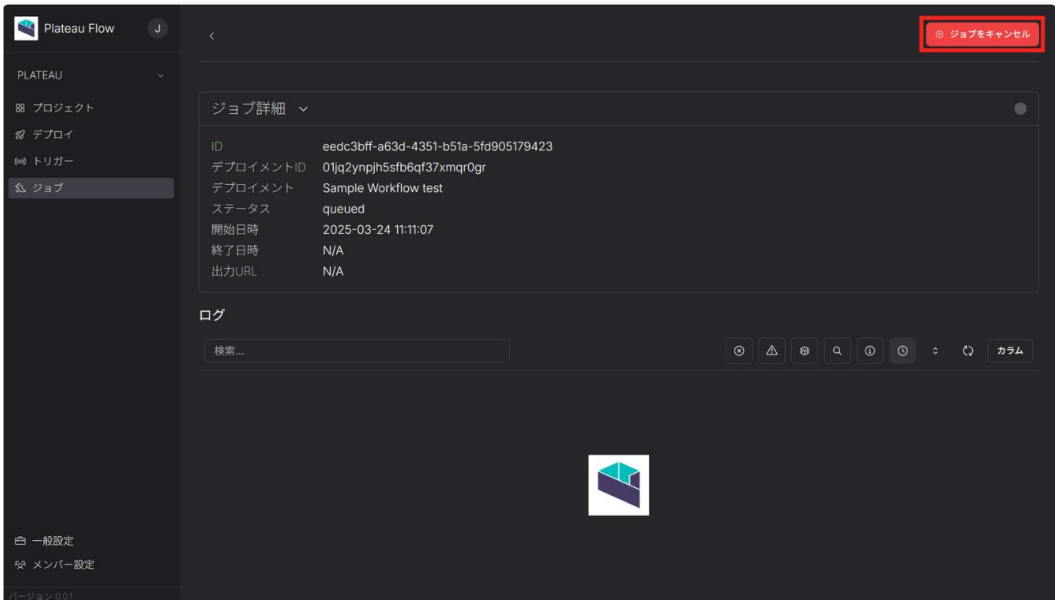
ジョブのログ
ジョブの詳細ページでは、実行ログを確認し、デバッグを行うことができる。
ログのフィルタリング機能
必要な情報を素早く見つけるために、フィルタリング機能を活用できる。

フィルタ機能	説明
エラー	エラーが発生したログのみを表示。
警告	注意が必要なログを表示。
デバッグ	プログラムやシステムの不具合を見つけて修正した情報を表示。
トレース	詳細な処理の流れを表示。
情報	ログ情報を表示。
タイムスタンプ	各ログの発生時間を表示。
リセットログ	フィルターをリセットし、すべてのログを表示。



ログのカラムをカスタマイズ
ログテーブルのカラムをカスタマイズすることで、必要な情報を整理し、デバッグを効率化できる。
- 「カラム」ボタンをクリックし、表示する項目を選択。
- 以下の項目を表示・非表示の切り替えが可能。
    - タイムスタンプ: 各ログの発生時間を表示。
    - ステータス:エラー、警告、情報などの重要度を表示。
    - メッセージ:ログの詳細情報を表示。

実行中のジョブのキャンセル
- ジョブが待機中（Queued）または実行中（Running）の場合、ジョブ詳細ページの右上にある「キャンセル」ボタンをクリックする。



2.3.3 その他の機能

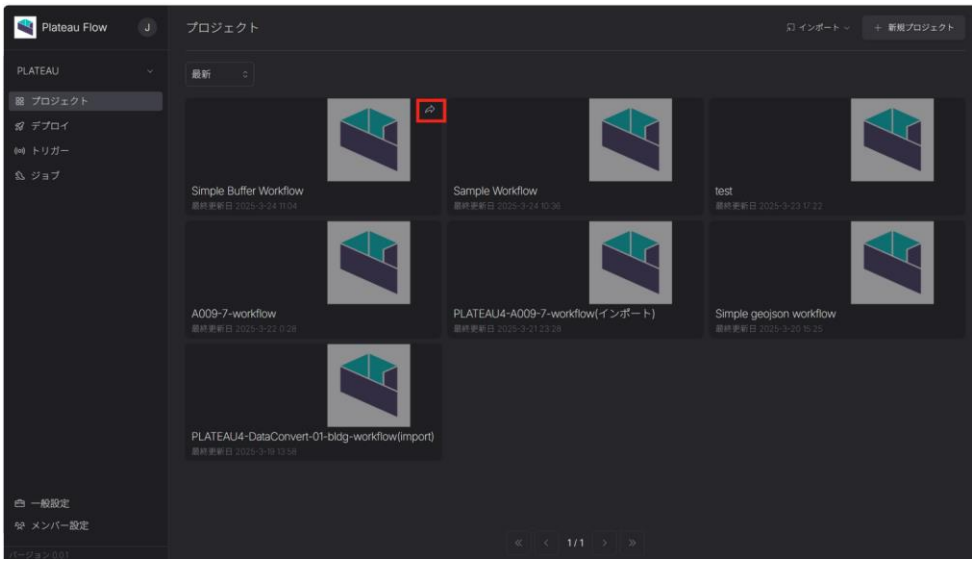
2.3.2 で説明したワークフローの基本操作に加え、Flow で利用可能な補助機能について紹介する。

（1）ワークフローのURL共有機能

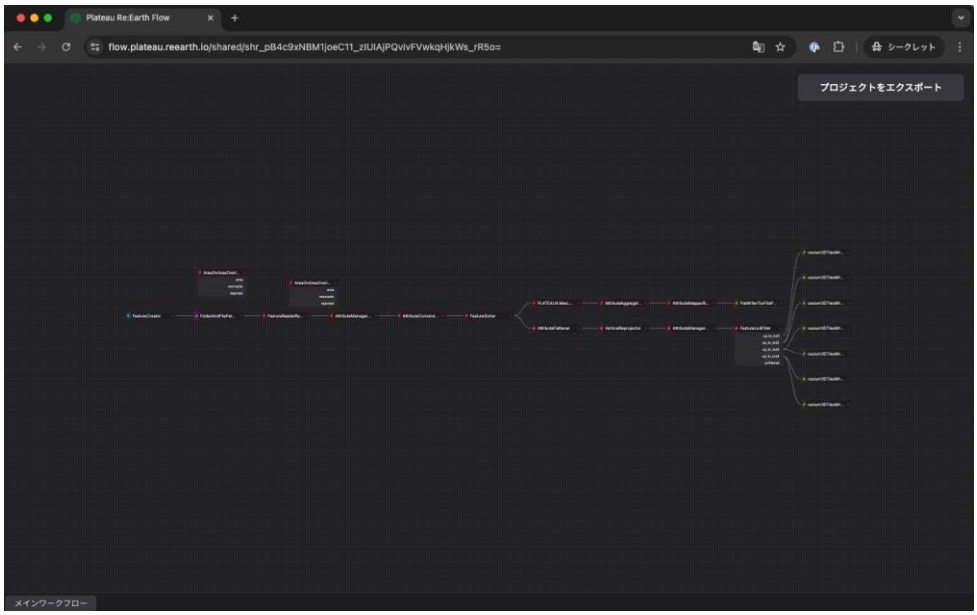
- ワークフローのURLリンクを他のユーザーと共有できる。

手順

1. ワークフローエディターのアクションパネル内の縦3点リーダーをクリックし「プロジェクトを共有」をひらいて、「共有」を選択して送信する。
2. 共有後、プロジェクト一覧に共有アイコンが表示されるようになる。



3. 対象プロジェクトの右下をクリックし、共有URLをコピーする。
4. 他のユーザーがこのURLをクリックすると、ワークフローを確認できる。



（2）プロジェクトのインポート / エクスポート機能

- 作成したプロジェクトをエクスポートし、他のワークスペースでインポートできる。

手順

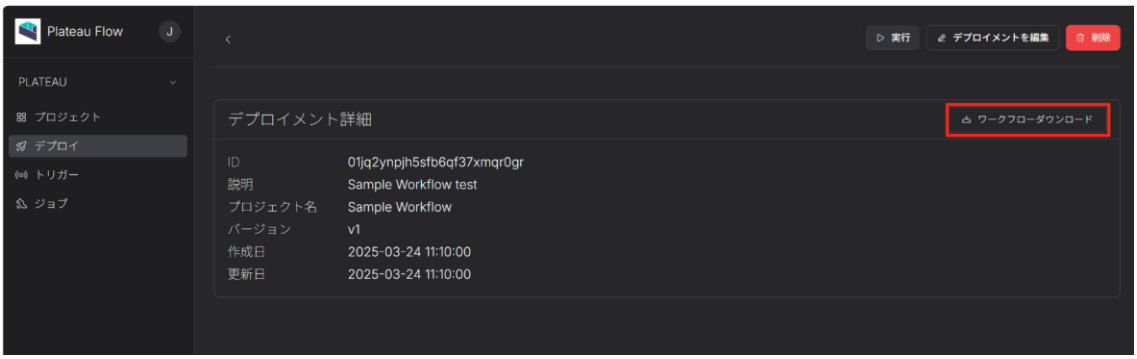
1. エクスポート: プロジェクト一覧ページから、作成済みのプロジェクトをエクスポート可能。
2. インポート: エクスポートしたプロジェクトを所有するワークスペースにインポート可能。

（3）デプロイ済みワークフローのエクスポート機能

- デプロイ済みワークフローをダウンロードできる。

手順

1. デプロイ詳細ページの右上にある「ワークフローをダウンロードする」ボタンをクリックする。



（4）バージョン管理機能

- プロジェクトの過去のバージョンを確認し、必要に応じて特定のバージョンに戻す。

手順

1. ワークフローエディターのアクションパネルから「バージョン履歴」を選択すると、右側に履歴が表示される。
2. 対象のバージョンを選択すると、そのバージョンの状態に戻すことができる。

2.3.4 簡単なワークフローを構築して実行する

本項では、Flow を使用して簡単なワークフローを構築し、実際にデータを処理する流れを解説する。

具体的には、GeoJSONデータの読み込み、バッファ（領域拡張）の適用、変換後データの出力までの一連の操作を行う。

この操作を通じて、ワークフローの基本構造や構築手順、実行手順の理解を深め、実際の活用方法を紹介する。

ワークフローを構築する

まずはワークフローエディター内でデータの処理手順を定義するワークフローを構築する。

ワークフローの主な構築手順

今回のワークフローは主に4つの手順で行う。

（1）新規プロジェクトの作成

- ワークフローを管理するためのプロジェクトを作成する。

（2）データを読み込む（FileReaderノードの追加）

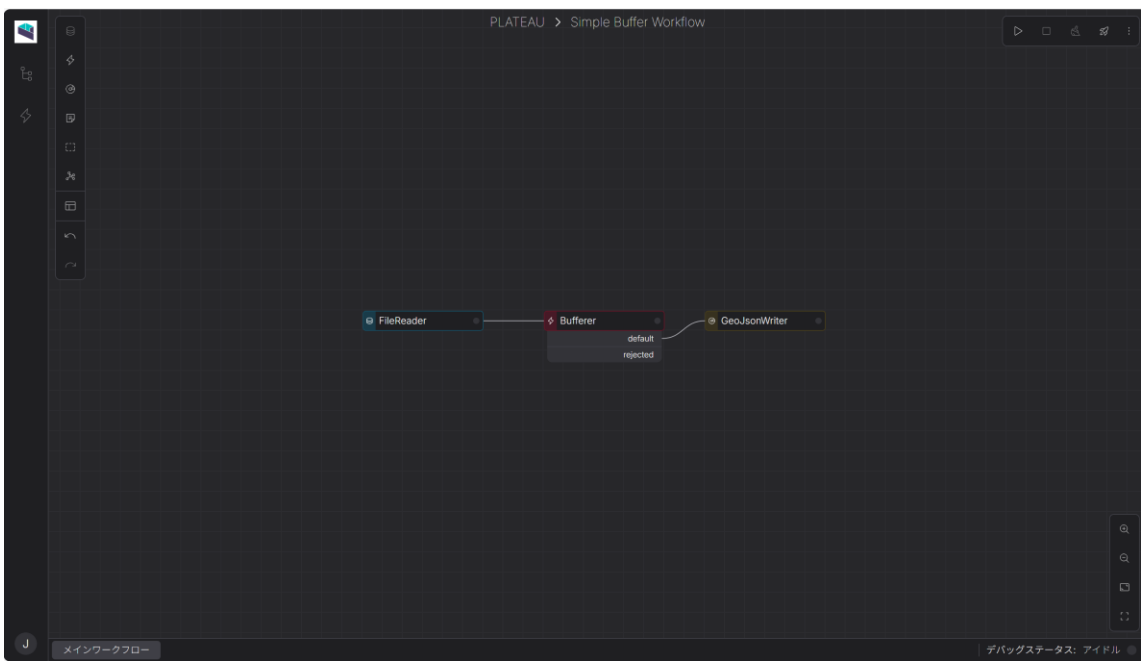
- FileReaderノードを使用して、GeoJSON形式の地理空間データをワークフローに取り込む。

（3）データを変換する（Buffererノードの追加・接続）

- Buffererノードを使用して、地図上のポリゴン（領域）を指定した距離だけ拡張する。

（4）変換後のデータを保存する（GeoJsonWriterノードの追加・接続）

- GeoJsonWriterノードを使用して、バッファ処理後のデータを新しいGeoJSONファイルとして保存する。



(

（1）新規プロジェクトの作成

1. PLATEAU Flow にログインする。

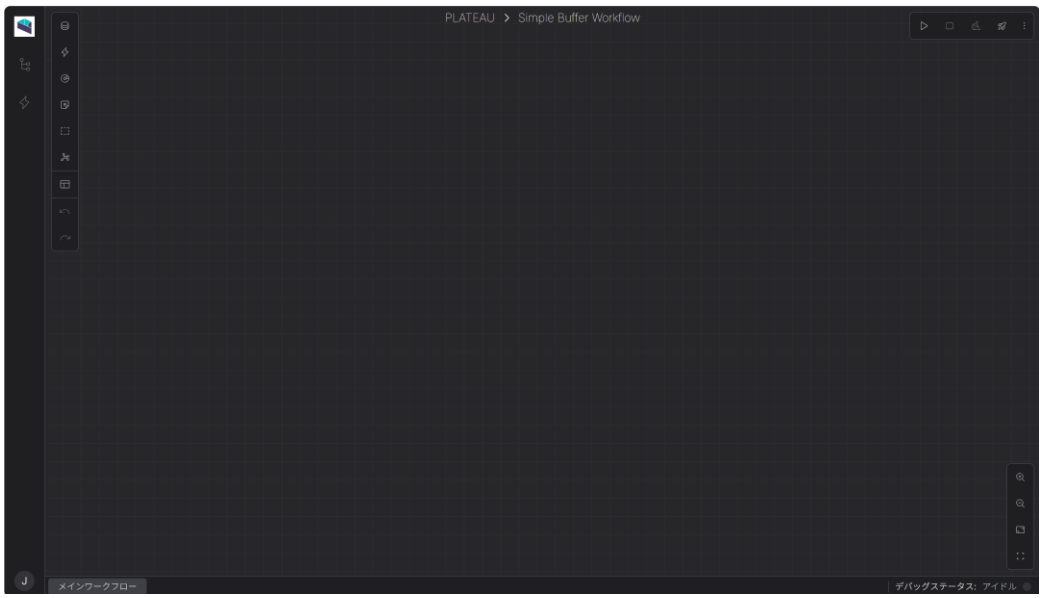
• ログイン後、ダッシュボード画面が表示される。

2. ダッシュボード左上の 「プロジェクト」 をクリックし、プロジェクト一覧を開く。
3. ダッシュボード右上の 「+ 新規プロジェクト」 ボタンをクリックする。
4. 表示されたダイアログにて、プロジェクト情報を入力する。

• プロジェクト名: 任意のプロジェクト名を入力する。（例:「Simple Bufferer Workflow」）
• プロジェクトの説明（オプション）: プロジェクトの目的や概要を記入する。

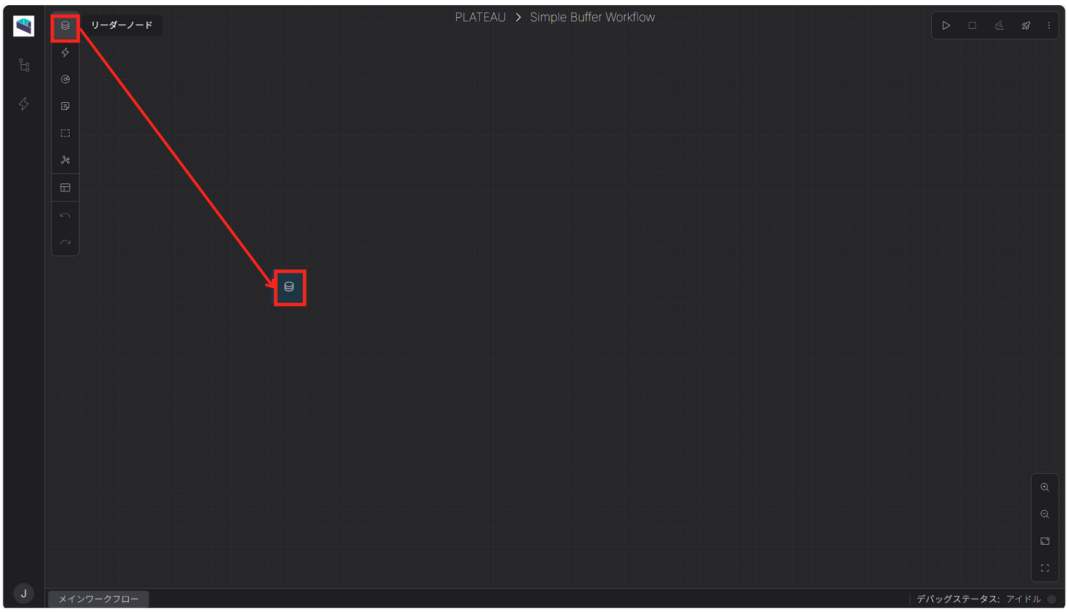
5. 「作成」ボタン をクリックすると、プロジェクトが作成される。
6. 作成後、プロジェクト一覧に表示され、プロジェクトをクリックするとワークフローエディターに移動する。

• (2)～(4)の作業はワークフローエディター上での操作となる。



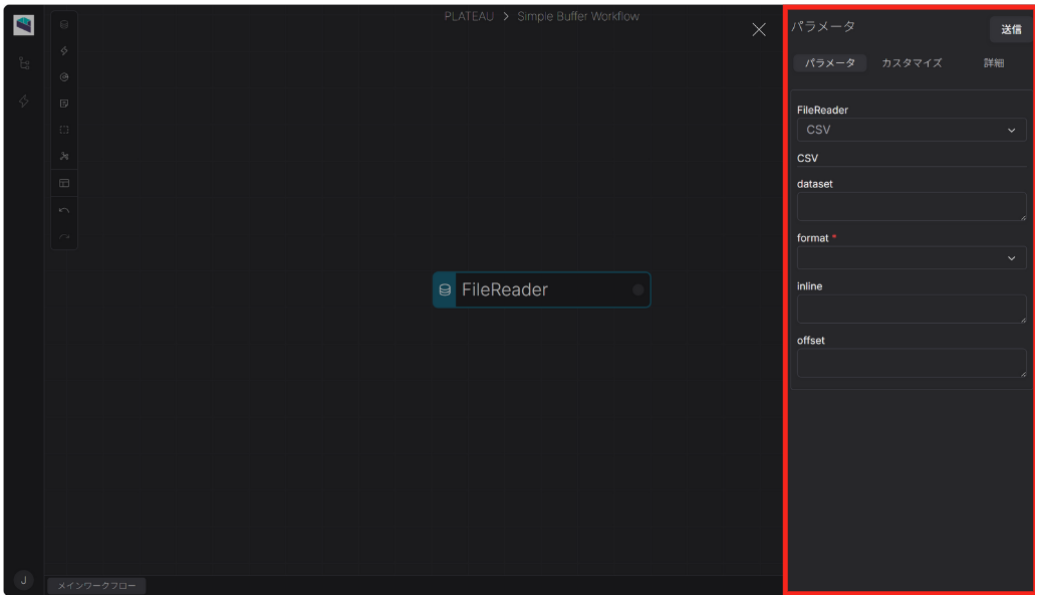
（2）データを読み込む（FileReaderノードの追加）

1. キャンバスツールバーから リーダーノードを キャンバスにドラッグ＆ドロップする。



2. ダイアログボックス が開くため、「FileReader」アクションを選択する。

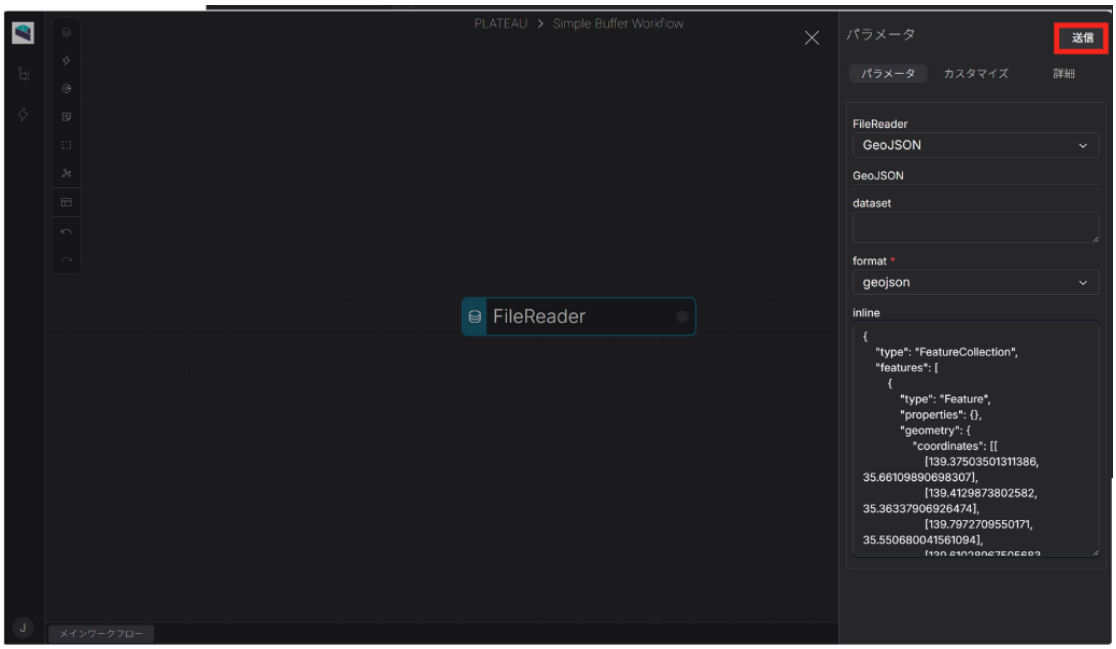
3. リーダーノードをダブルクリックして、パラメータを表示させる。



4. パラメータにに以下の値を入力する。
    - FileReader: GeoJson
    - dataset: 空欄のまま
    - format: geojson
    - inline: （以下のデータをコピー＆ペーストしてください）

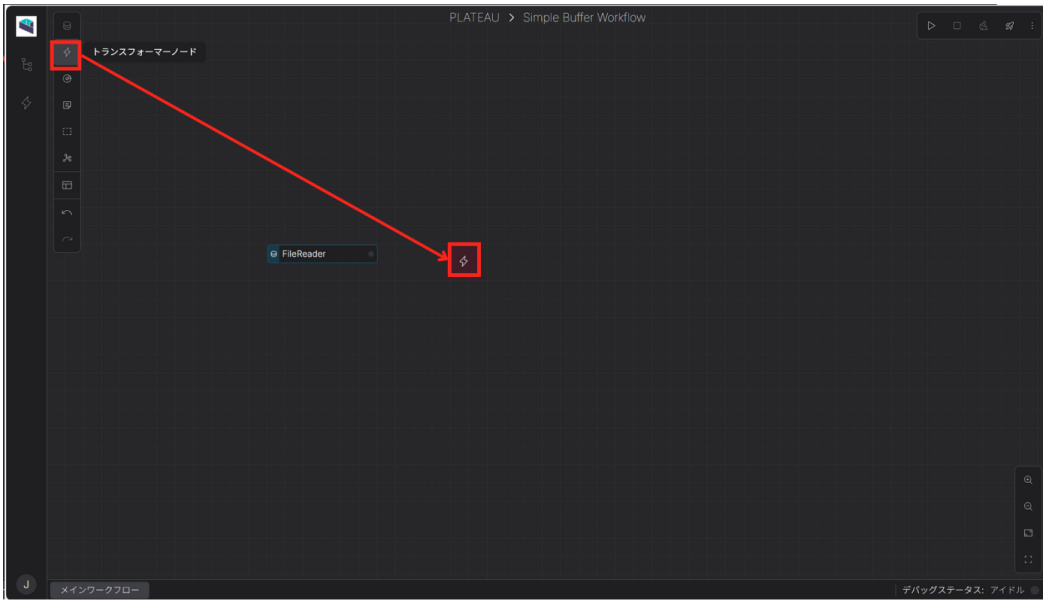
```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {},
      "geometry": {
        "coordinates": [[
          [139.37503501311386, 35.66109890698307],
          [139.4129873802582, 35.36337906926474],
          [139.7972709550171, 35.550680041561094],
          [139.61028967505683, 35.808552077864945],
          [139.55236396451136, 35.790477210623706],
          [139.52857932482067, 35.739908813339284],
          [139.37503501311386, 35.66109890698307]
        ]],
        "type": "Polygon"
      }
    }
  ]
}
```

5. 画面右上の「送信」ボタンをクリックして、パラメータを保存する。



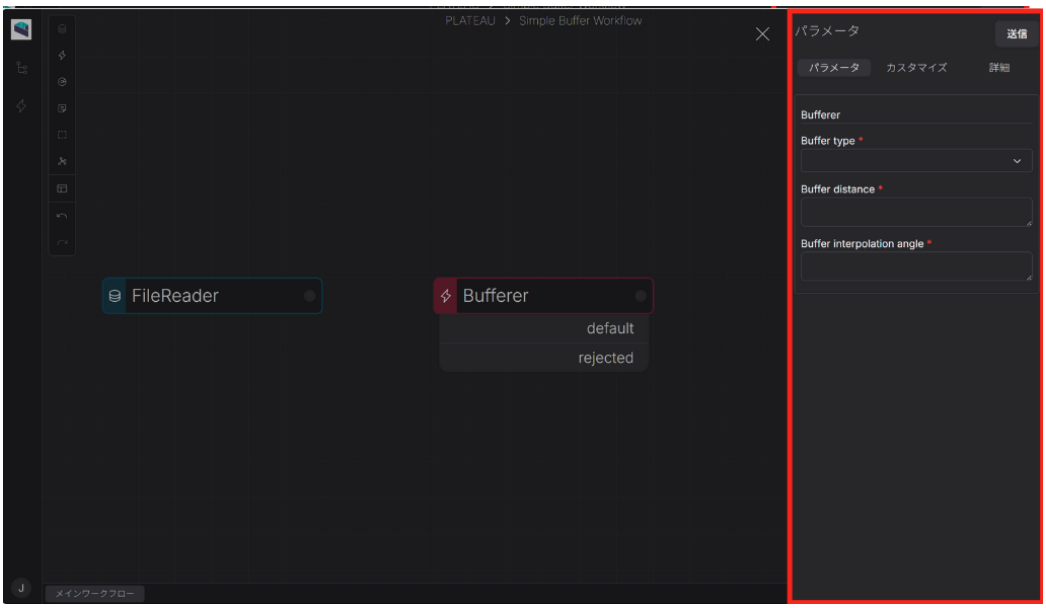
（3）データを変換する（Bufferer ノードの追加・接続）

1. キャンバスツールバーからトランスフォーマーノード（Transformer）をキャンバスにドラッグ＆ドロップする。

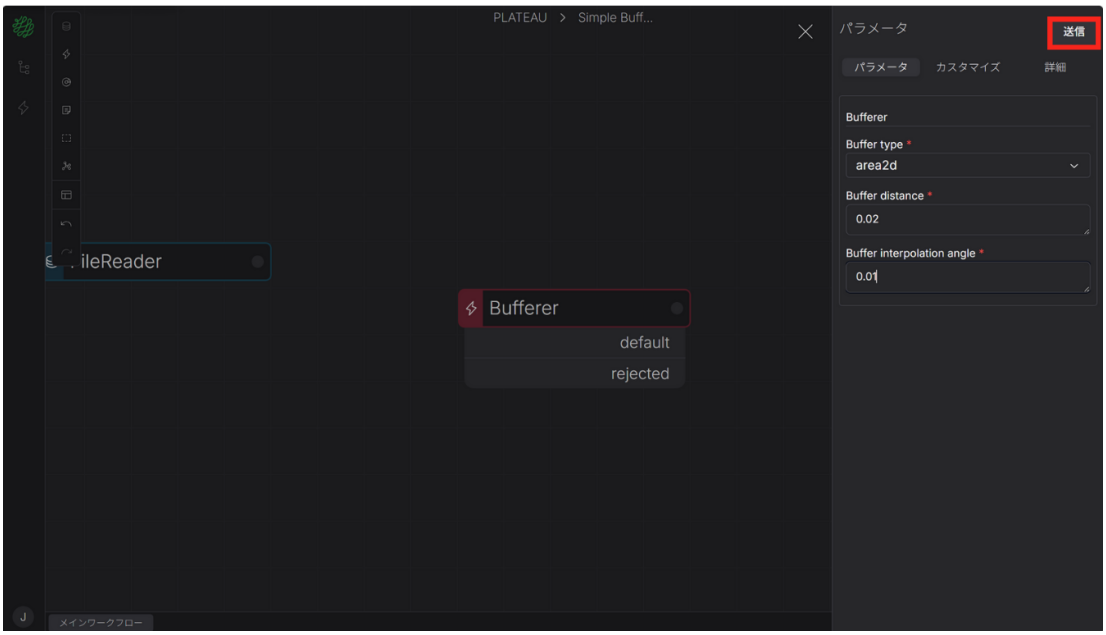


2. ダイアログボックスが開くため、「Bufferer」アクションを選択する。

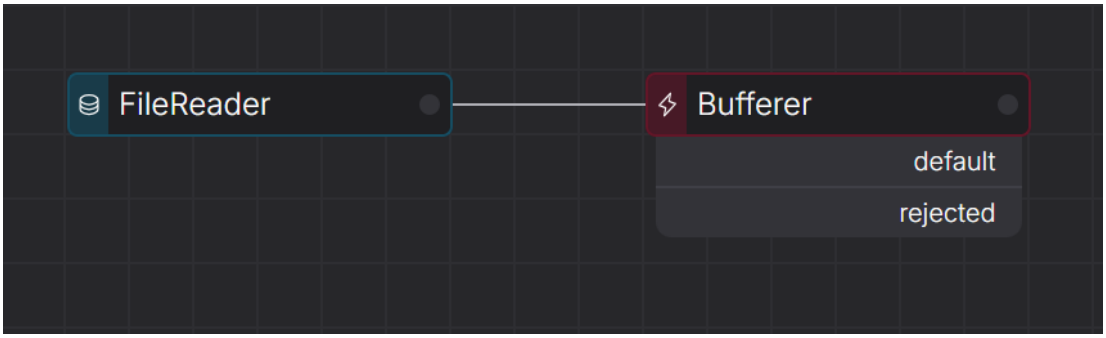
3. トランスフォーマーノードをダブルクリックして、パラメータを表示させる。



4. パラメータに以下の値を入力する。
    - Buffer Type: area2d
    - Buffer Distance: 0.02
    - Buffer Interpolation angle: 0.01
5. 画面右上の「送信」ボタンをクリックしてパラメータを保存する。



6. 「FileReader」ノードの右端をクリックすると、接続用のコードが表示される。このコードをドラッグして「Bufferer」ノードに接続する。

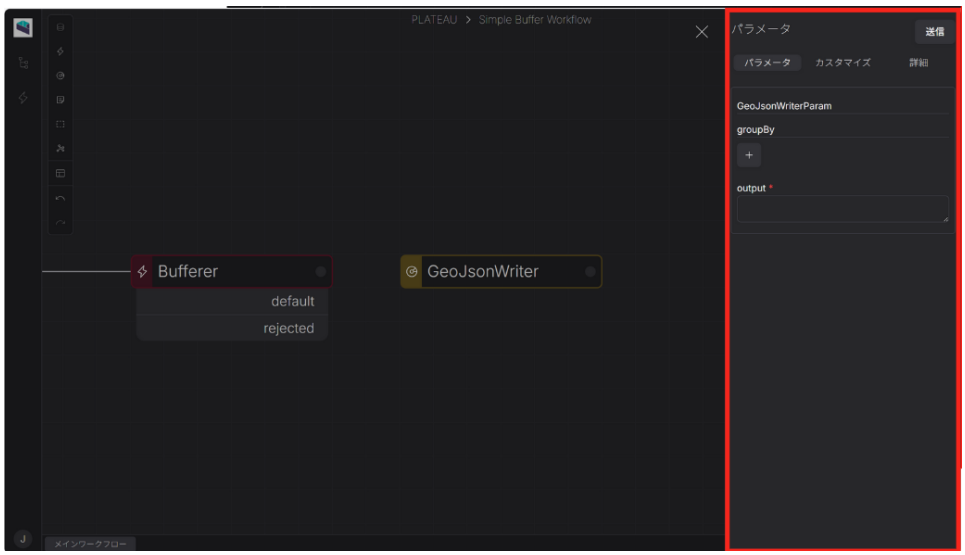


（4）変換後のデータを保存する（GeoJsonWriterノードの追加・接続）

1. キャンバスツールバーからライターノード（Writer）をキャンバスにドラッグ＆ドロップする。

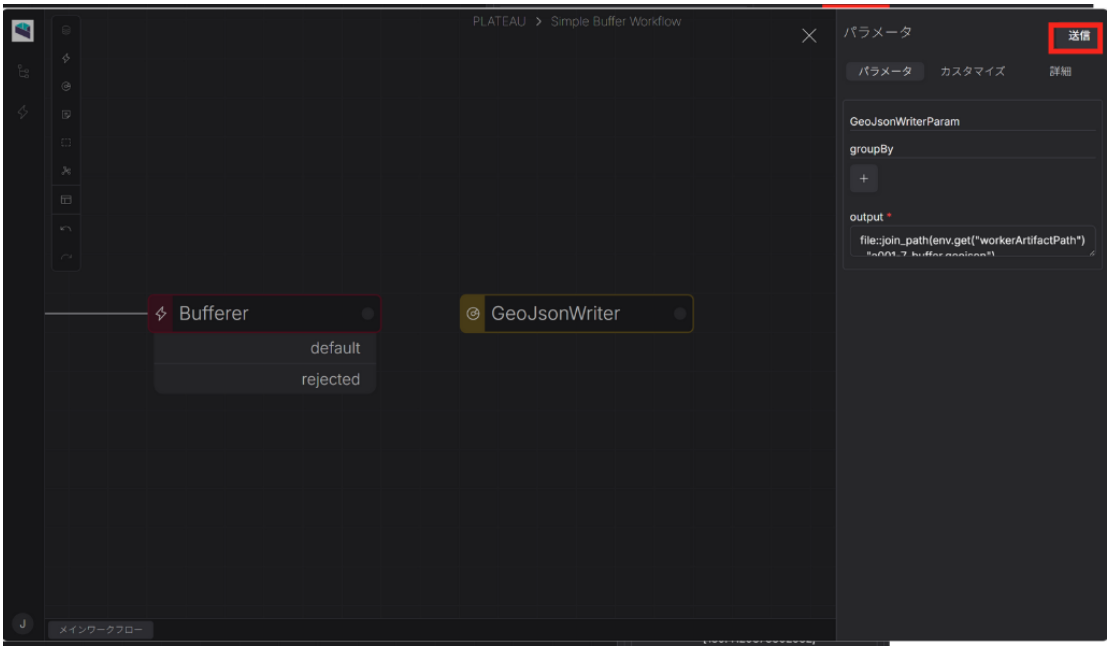
2. ダイアログボックスが開くため、「GeoJsonWriter」アクションを選択する。

3. ライターノードをダブルクリックして、パラメータを表示させる。

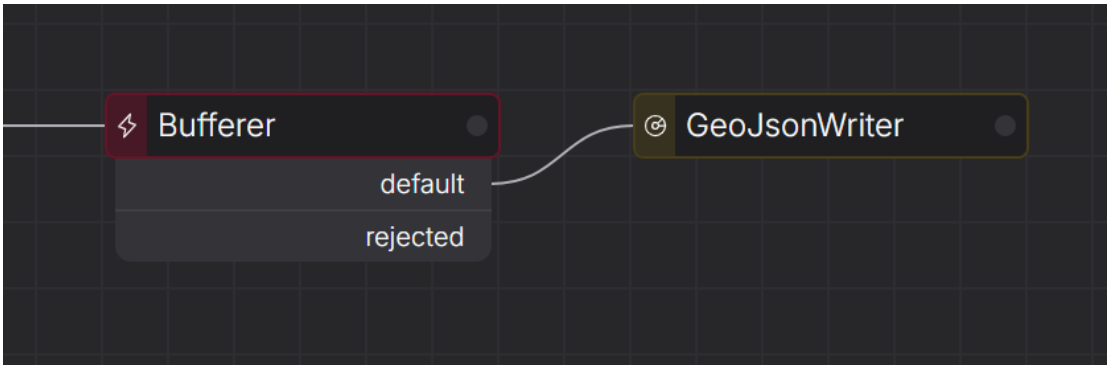


4. パラメータに以下の値を入力する。
    - groupBy: 空欄のまま
    - output: file::join_path(env.get("workerArtifactPath"), "a001-7-buffer.geojson")

5. 画面右上の「送信」ボタンをクリックしてパラメータを保存する。



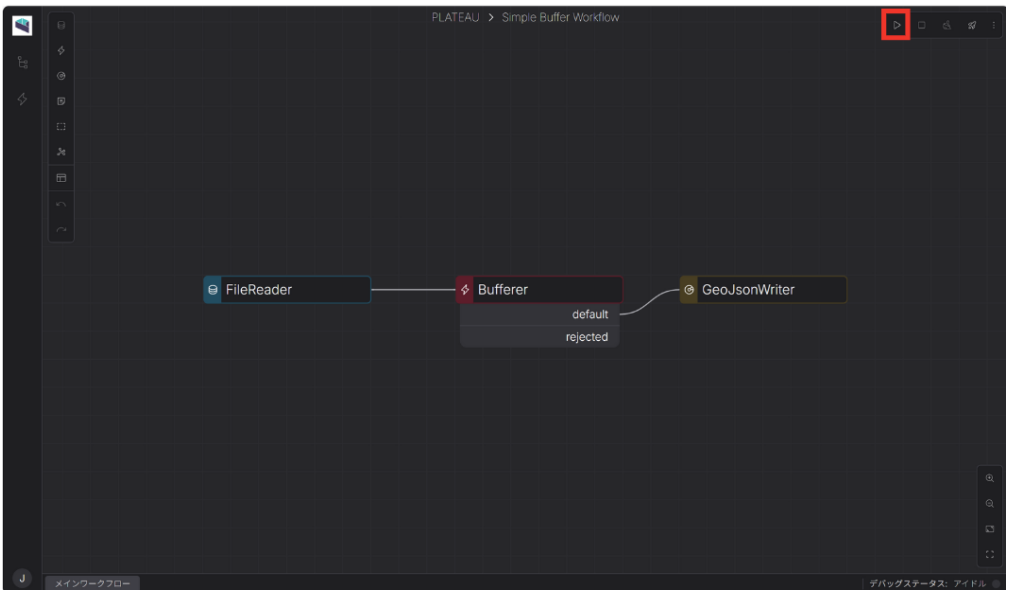
6. 「Bufferer」ノード内「default」の右端をクリックすると、接続用のコードが表示される。このコードをドラッグして「GeoJsonWriter」ノードに接続する。



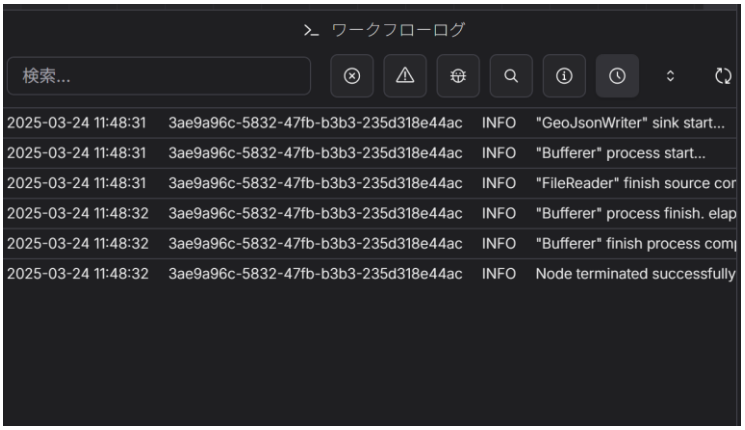
ワークフローを実行する

次に、構築したワークフローをデプロイし、実際に実行する手順について説明する。
ワークフローをデプロイすることで、処理を実行できる環境が整い、データの変換や分析を行うジョブを作成・管理できるようになる。
以下の手順に従い、ワークフローを正しく動作させ、処理結果を確認するまでの操作方法を進めていく。

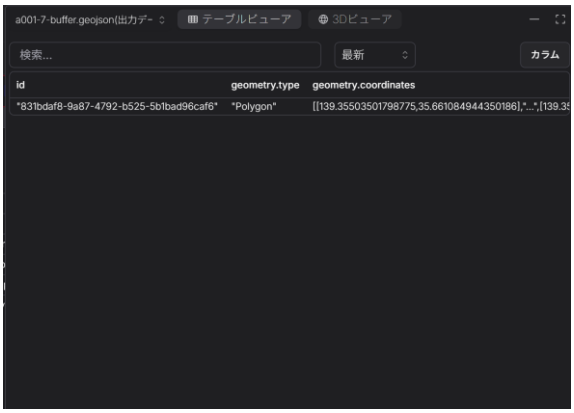
1. ワークフローエディター上で動作確認を行うため、アクションパネル内の「ワークフローのデバッグ実行を開始」ボタンをクリックする。



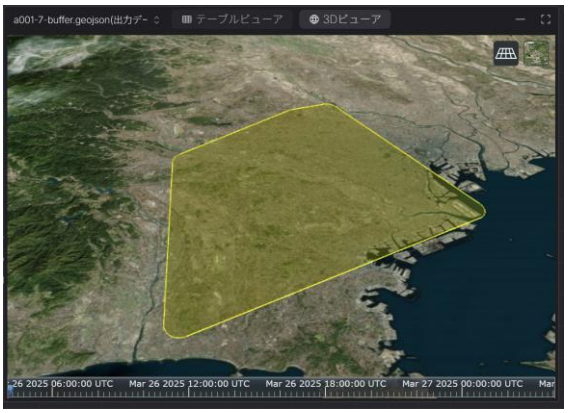
2. デバッグ実行後、ワークフローログ、テーブルビューア、3Dビューアを使用して、処理結果に誤りがないかを確認する。
    - ワークフローログ:ワークフロー実行時の各処理ステップの状態やエラー内容、実行時間などを確認できるログ表示機能。



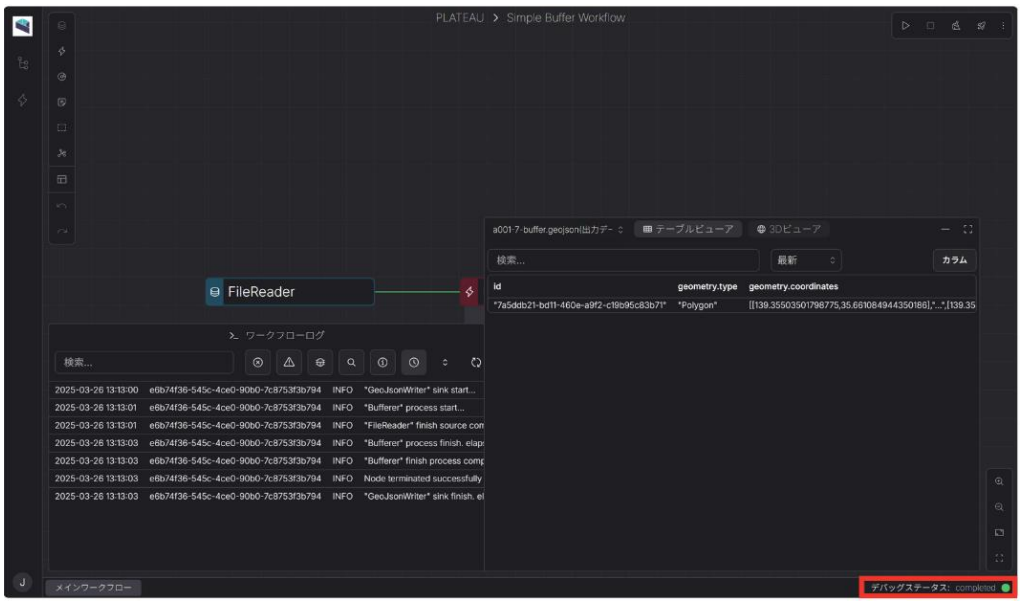
- テーブルビューア:関数や処理ステップの入出力データを表形式で可視化し、データの構造や値を確認できる機能。



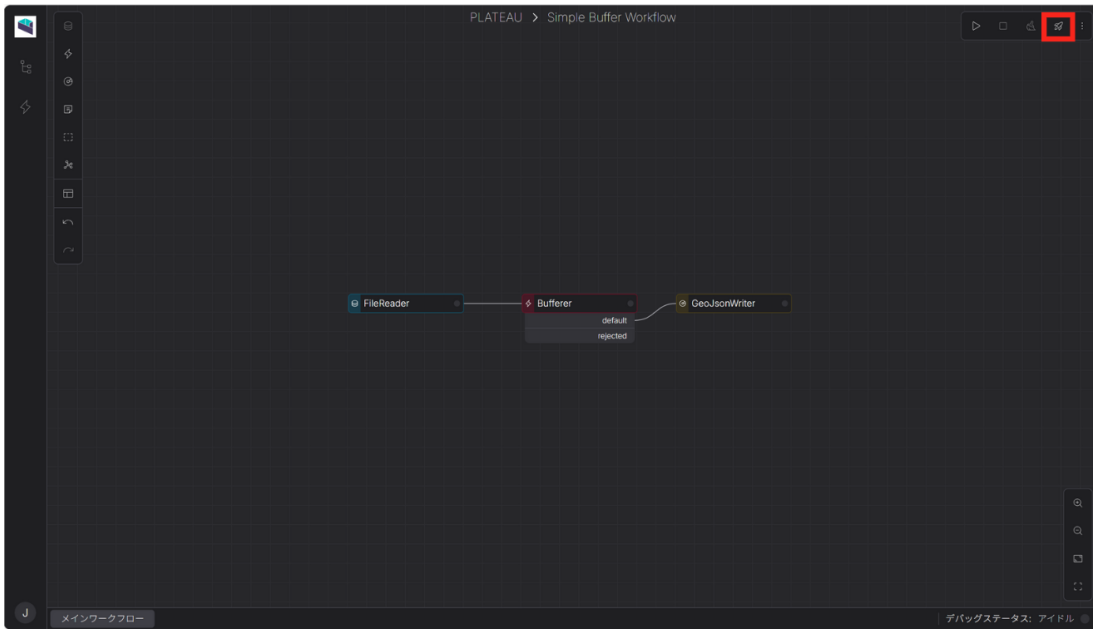
- 3Dビューア:対象地域や地物の位置・形状を直感的に把握できる機能。



3. 画面右下にあるデバッグステータスが「complete」になっており、ワークフロー全体がエラーなく正常に実行されたことを確認する。



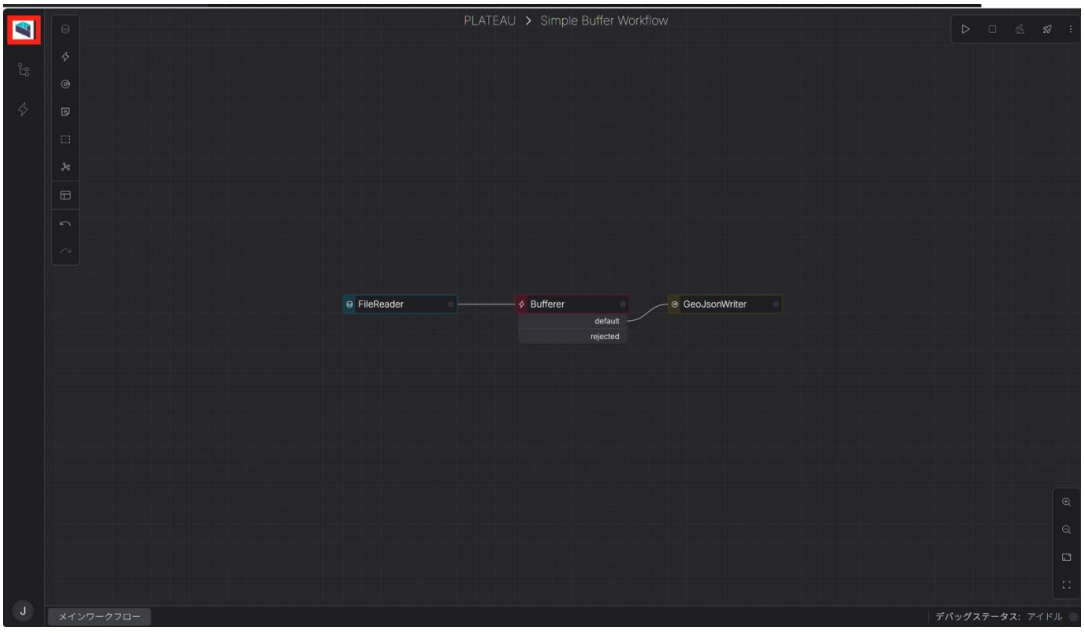
4. ワークフローエディター画面右上のアクションパネルから「デプロイ」ボタンを選択する。



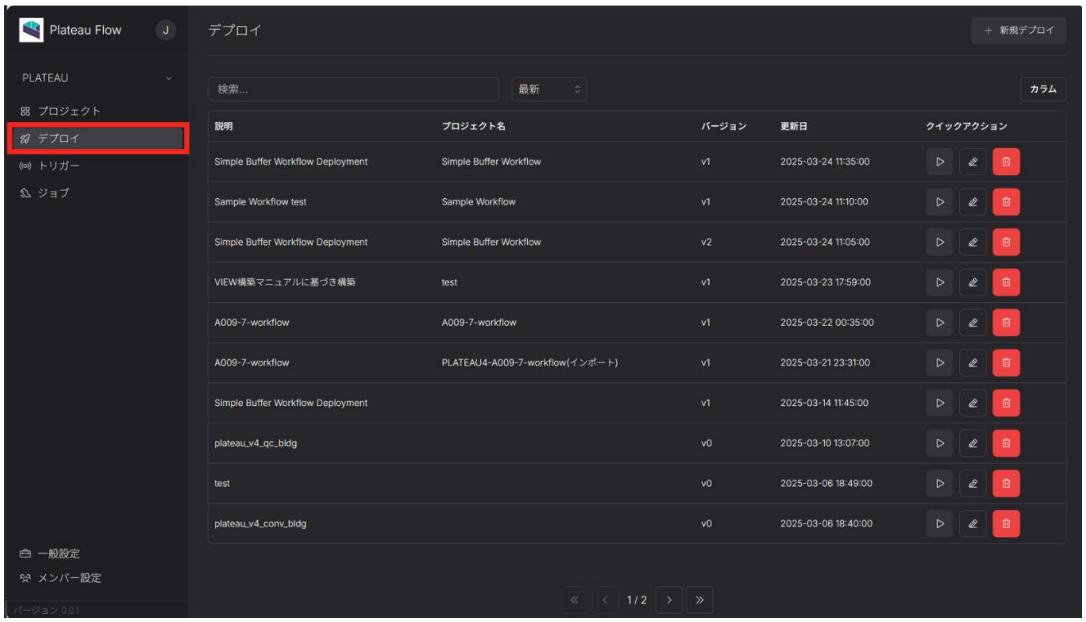
5. ダイアログボックス内の「デプロイメントの説明」欄を入力する。（例：「Simple Buffer Workflow Deployment」）

6. 入力が完了したら、「デプロイ」ボタンをクリックしてワークフローをデプロイする。
- デプロイの作成が完了すると、画面右下に「デプロイメント作成完了」という通知が届く。

7. 画面左上にあるFlowアイコンをクリックし、ダッシュボードに戻る。

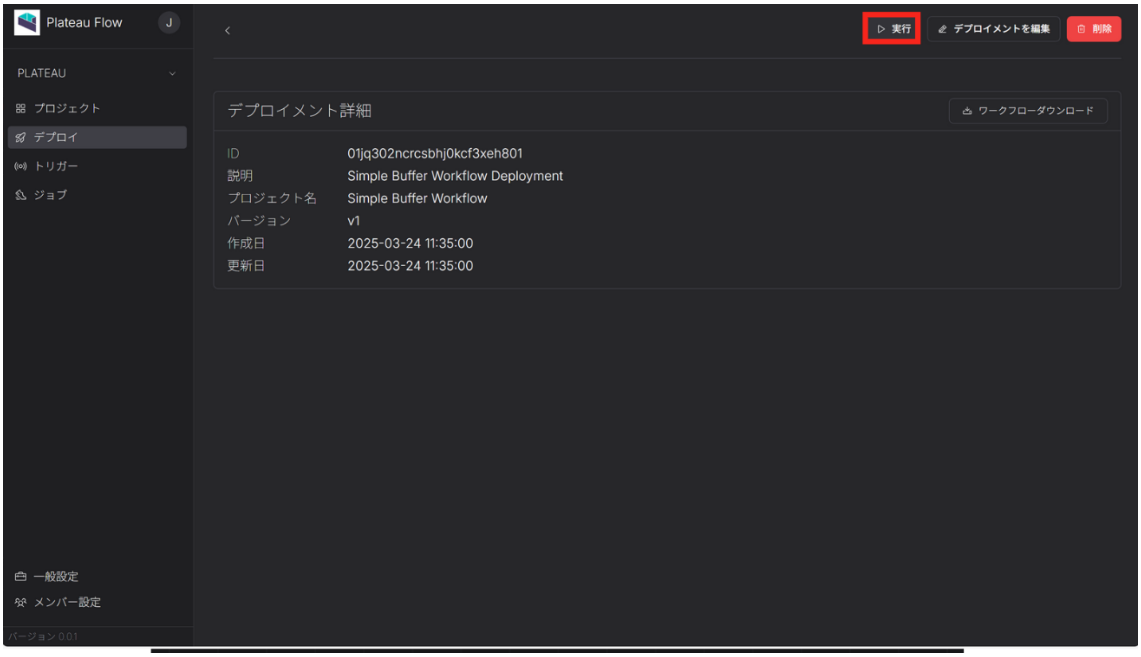


8. ダッシュボード画面左サイドパネルから「デプロイ」タブを開く。

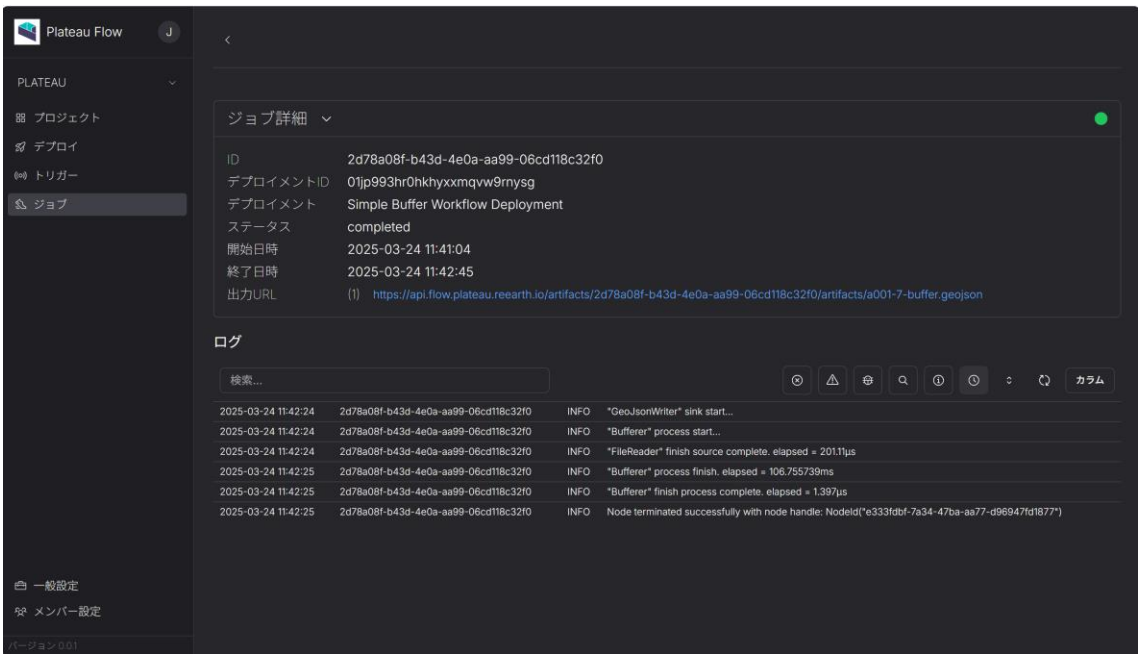


9. 新たに作成されたデプロイメントをクリックし、「デプロイメント詳細」ページに移動する。

10. 「実行」ボタンをクリックして、ワークフローを実行する。
- 実行を開始すると、画面右下に「デプロイメント実行完了」という通知が届く。



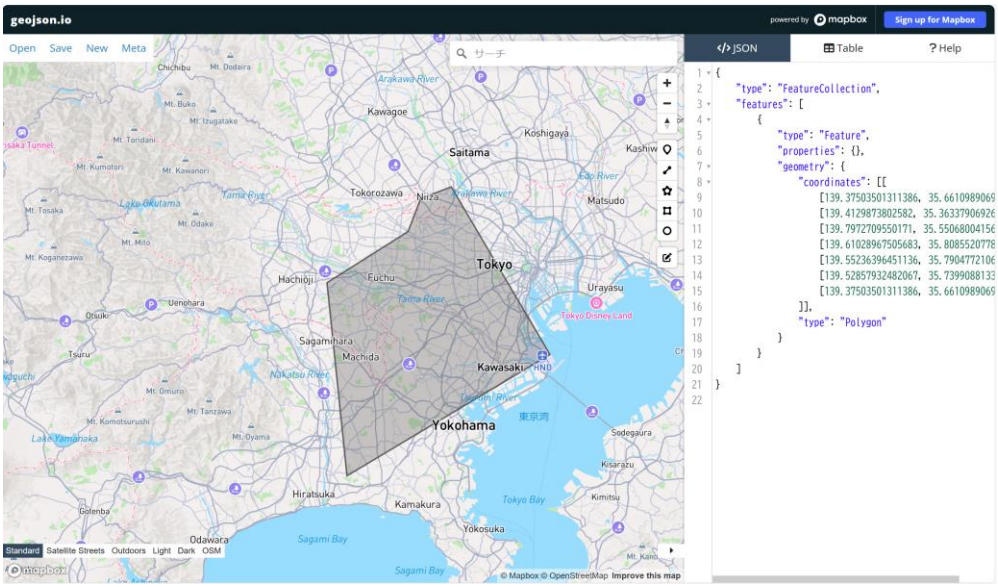
11. デプロイ済みワークフローが実行されるとジョブが作成され、自動的に「ジョブ詳細」ページに移動する。
12. ジョブの ステータスやログを確認し、完了後に 出力データのURLを取得できる。



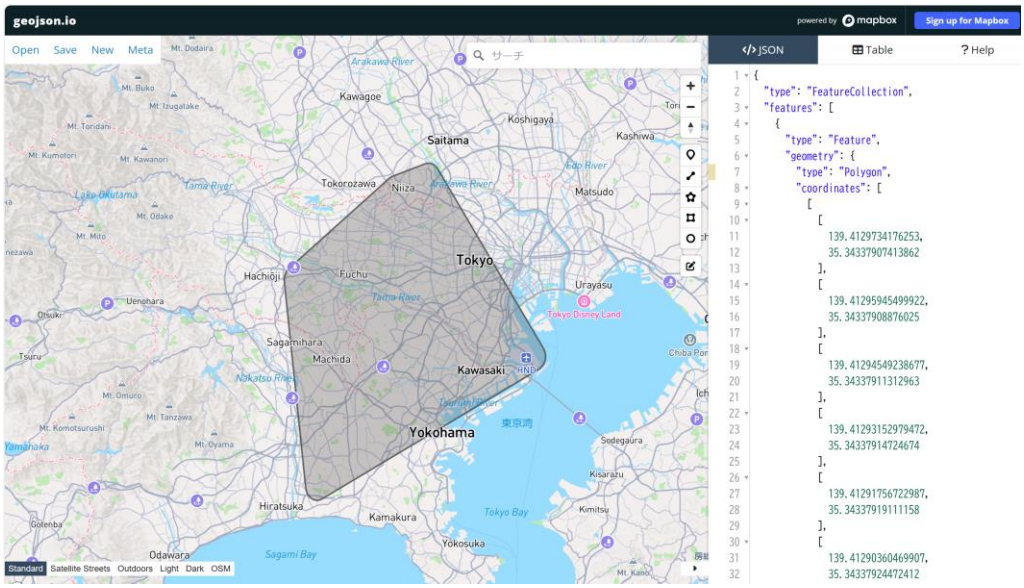
データの比較（バッファ適用前後）

出力URLを取得し、変換後のデータをgeojson.ioで確認すると、バッファ処理の適用により、ポリゴンの領域が拡張され、形状が滑らかになったことが確認できる。

- **バッファ適用前**



- **バッファ適用後**



その他、対応している空間演算例

バッファ（Buffer）

指定した距離で対象物の周囲に新しいポリゴンを生成する

Before

入力データ：点、線、ポリゴン

After

出力データ：元の地物を包含する新しいポリゴン。距離に応じてサイズが変化

切り抜く（Clip）

指定したエリア内の地物のみを抽出する

Before

入力データ：対象レイヤー + クリップ用ポリゴン

After

出力データ：クリップ用ポリゴン内に存在する対象レイヤーの地物のみ

凸包（Convex Hull）

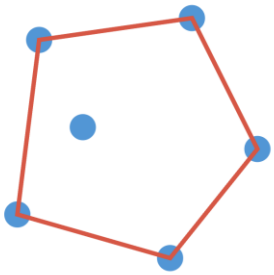
点群を囲む最小の凸多角形を生成する

Before

入力データ：点の集合

After

出力データ：すべての点を含む凸多角形

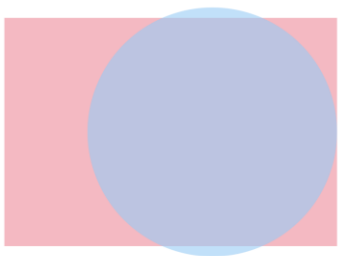


差分（Difference）

第1レイヤーから第2レイヤーと重なる部分を除去する

Before

入力データ：2つのレイヤー



After

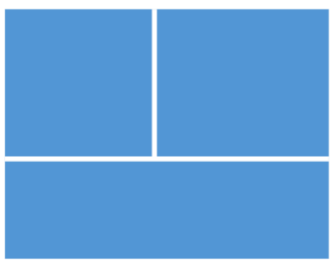
出力データ：第1レイヤーから第2レイヤーと重なる部分を除いた地物

融合（Dissolve）

指定した属性に基づいて隣接する地物を融合する

Before

入力データ：単一レイヤー



After

出力データ：指定した属性値が同じ隣接地物が融合された、より単純化されたレイヤー

交差（Intersect）
複数のレイヤーが重なり合う部分のみを抽出する

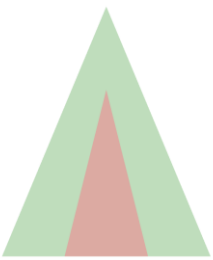
Before
入力データ：すべての入力レイヤーが重なり合う部分のみの新しいレイヤー

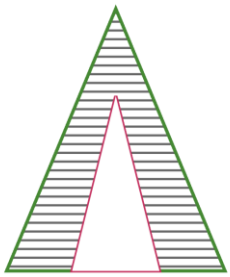
After
出力データ：すべての入力レイヤーが重なり合う部分のみの新しいレイヤー

対称差（Symmetrical Difference）
2つのレイヤーの重複しない部分のみを抽出する

Before
入力データ：2つの入力レイヤーの重複しない部分のみの新しいレイヤー

After
出力データ：2つの入力レイヤーの重複しない部分のみの新しいレイヤー

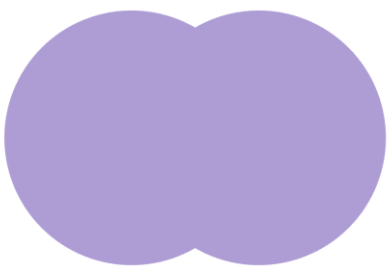




和集合（Union）
複数のレイヤーを1つに統合する

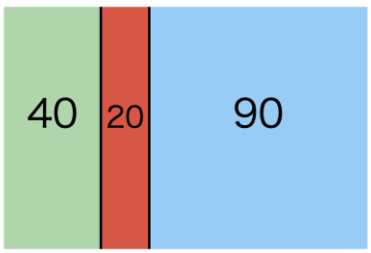
Before
入力データ：複数のレイヤーを1つに統合する

After
出力データ：すべての入力レイヤーの地物を含む単一の新しいレイヤー。属性テーブルはすべての入力を保持

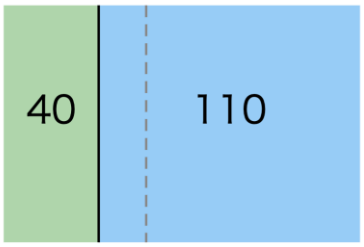


選択物の隣接ポリゴンの融合（Eliminate）※最大面積ポリゴンとの融合
選択した小さなポリゴンを隣接する大きなポリゴンに融合する

Before
入力データ：選択した小さなポリゴンを隣接する大きなポリゴンに融合する



After
出力データ：選択したポリゴンが隣接する最適なポリゴンに融合された結果



第３章　PLATEAU VIEW 4.0の環境構築

3.1 PLATEAU CMS / Editor / Flowの環境構築

3.1.1 システム構成の全体像

本章では、システム構築を担当するエンジニアを想定読者として、CMS・Editor・Flowのシステムの構築をGoogle Cloud上で行う。それぞれのシステムの全体像は1.1を参照すること。なお、PLATEAU VIEW 4.0（主にPLATEAU Flow）については、AWS環境でのシステム構築に対応していない。PLATEAU VIEW 3.0はAWS環境での構築が可能であるため、AWS環境での構築を希望する場合は、「PLATEAU VIEW構築マニュアル（第4.0版）」を参照されたい。

3.1.2 各種サービスのセットアップ

前項で示したシステムを構築するためのセットアップ手順を示す。事前に以下の準備が必要である。

- クレジットカード（Google Cloudなどで使用）
- 本システムで使用するドメイン（管理画面にアクセスしてネームサーバーの変更ができること）

本項で示す各種コマンドは、特に注記がない限り、Windows（PowerShell）とMacOS/Linuxの両方に対応している。注記がある場合はそれに従うこと。

（1）Auth0のセットアップ

Auth0（https://auth0.com/jp）にアクセスしてAuth0のアカウントを作成し、テナントを開設する。ドメイン名・リージョンは問わない。なお（2025年3月現在）東京リージョンを選ぶことができる。

次に、公式のQuick Startを参考に、Auth0 Management APIのセットアップを行う（「Configure the Provider」の項の前まで）。

https://github.com/auth0/terraform-provider-auth0/blob/main/docs/guides/quickstart.md

注意：PLATEAU Editorなどへのログインに失敗する原因になるため、今この時点ではまだ自身のユーザーをAuth0のテナント内に作成しないこと。

後のステップで必要となる情報は以下のとおりなので確認しておくこと。

- ドメイン
- クライアントID
- クライアントシークレット

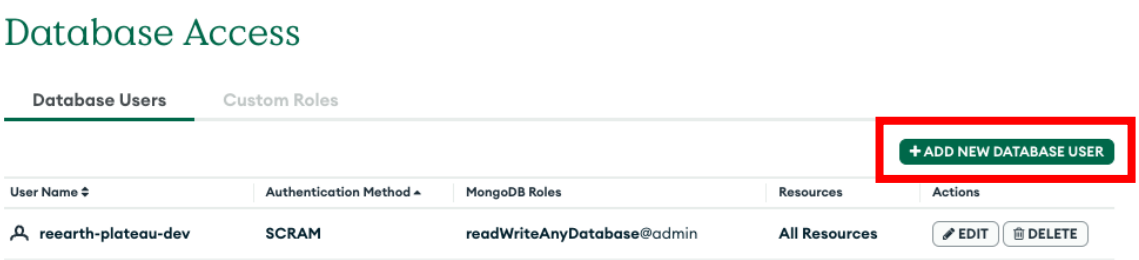
後で使用するので以下のコマンドを実行して、クライアントシークレットを変数に出力しておく。

```
export AUTH0_CLIENT_SECRET="<クライアントシークレット>"
```

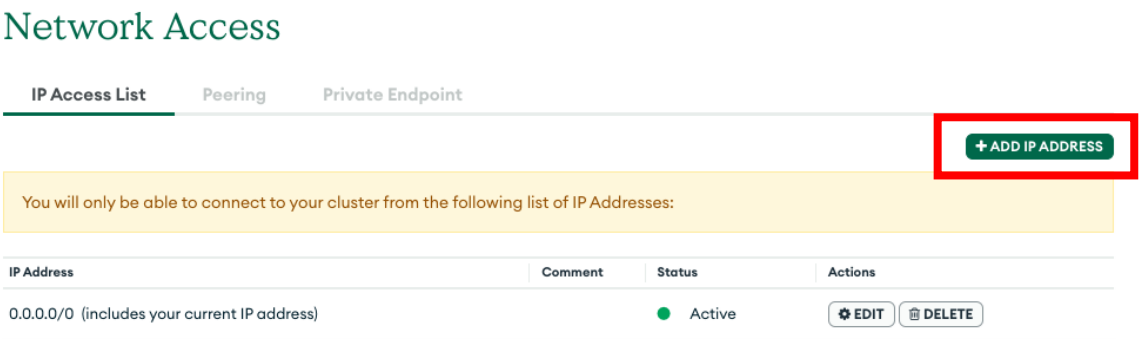
（2）MongoDB Atlas のセットアップ

MongoDB Atlas （https://www.mongodb.com/ja-jp/atlas） にアクセスしてアカウントを作成し、プロジェクト及びクラスタを作成する。なお有料のクラスタを作成する場合は、クレジットカードの登録が必要である。名前やリージョンなどは自由。

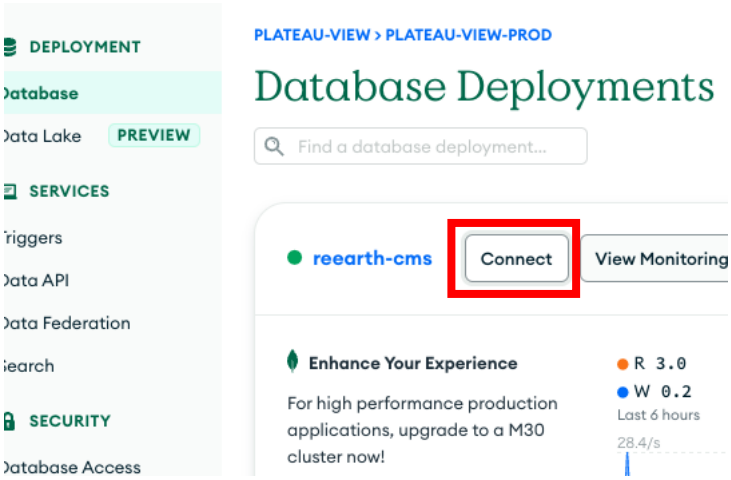
プロジェクトの「Database Access」の設定を確認し、データベースのユーザーが作成されていることを必ず確認する。作成されていない場合は、「ADD NEW DATABASE USER」ボタンを押下して、任意のパスワードなどの認証方式でユーザーを作成する。そのユーザーに対して、少なくとも読み書き（readWrite）の権限を許可するロール（Read and write to any database・Atlas Adminなど）を設定する必要がある。



次に、プロジェクトの「Network Access」の設定を確認し、全てのIPアドレスからの接続が許可されていることを必ず確認する。許可されていない場合は、「ADD IP ADDRESS」ボタンを押下し、Access List Entryで「0.0.0.0/0」と入力してエントリーを追加する。



クラスタを作成後、クラスタ名の右にある「Connect」ボタンを押下し、「Connect your application」からデータベース接続URLを取得する。なおデータベース接続URLは慎重に扱い、他人には共有しないこと。



後で使用するので以下のコマンドを実行して、データベース接続URLを変数に出力しておく。URL中の <password> を先ほど作成したユーザーのパスワードに置き換えることを忘れないよう注意すること。

```
export REEARTH_DB="<データベース接続URL>"
```

3.1.3 Google Cloud向けセットアップ

次に、Google Cloudを利用して環境構築する手順を記載する。

（1）事前準備

Google Cloud（https://console.cloud.google.com）にアクセスし、Googleアカウントでログインし、プロジェクトを作成する。プロジェクトにはあらかじめクレジットカードを登録し紐付けておくことが必要である。
後で使用するので以下のコマンドを実行して、プロジェクトIDを変数に出力しておく。

```
export PROJECT_ID="<Google CloudプロジェクトID>"
```

（2）ドメイン・プレフィックス名の決定

今回構築するシステムで使用したいドメインを変数に出力しておく。後のステップでドメインのDNS設定を変更するため、あらかじめドメインの確保を済ませておくこと。

例：plateauview.example.com

```
export DOMAIN="<ドメイン>"
```

次に今回構築するシステムで使用するプレフィックス名を決め、変数に出力しておく。

プレフィックス名とは、Google Cloud内にさまざまなリソースを作成する時の名称の先頭に付加される任意の文字列である。20文字以内で半角英数ハイフンが使用可能。例えばステージング環境と本番環境で変えるなどの用途がある。管理者にしか見えないので自由に決めて良く、Google CloudプロジェクトIDと同じにしても問題ない。

例：plateauview-test

```
export SERVICE_PREFIX="<プレフィックス名>"
```

（3）Google Cloudloud コマンド

以下の説明に従ってGoogle Cloudloudコマンドのインストールを行う。

https://cloud.google.com/sdk/docs/install?hl=ja

インストール後、ターミナル上でGoogle Cloudloudコマンドを使用してGoogleアカウントでログインを行い、初期設定を済ませる。

```
Google Cloudloud auth login --update-adc
```

ブラウザが開くのでGoogleアカウントでログインする。 ログインが完了したあとに、以下のコマンドを実行してプロジェクトを設定する。

```
Google Cloudloud config set project $PROJECT_ID
```

（4）Terraform コマンド

以下の手順に従い、Terraformコマンドをインストールする（v1.11.1で検証済み）。

https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

（5）Terraform変数ファイルの用意

GitHubの以下のURLからZipファイルをダウンロードして解凍し、terraform/Google Cloudへ移動する。

https://github.com/Project-PLATEAU/PLATEAU-VIEW-4.0/archive/refs/heads/main.zip

（6）Terraformバックエンドの作成

Terraformの状態（tfstate）を保存する場所のことをバックエンドと呼ぶ。ここでは先ほど作成した Google CloudプロジェクトにGoogle CloudS（Cloud Storage）バケットを作成する。

```
Google Cloudloud storage buckets create gs://<バケット名>
```

作成したバケットのストレージクラスおよびロケーションをgoogle_storage_bucket.tfに設定する。
以下の例では、ストレージクラス STANDARD およびロケーション ASIA に設定している。

```
resource "google_storage_bucket" "terraform" {
  location      = "ASIA"
  name          = var.Google Clouds_bucket
  storage_class = "STANDARD"
}
```

また、作成したバケットの名前を terraform.tf の backend の bucket に設定する。

```
terraform {
  backend "Google Clouds" {
    bucket = "<作成したバケット名>"
  }
  ...
}
```

そして、作成したGoogle CloudSバケットを取り込む。

```
# 初回一回のみ
terraform init

# APIの有効化
terraform import google_storage_bucket.terraform <バケット名>
```

（7）Google Cloud API の有効化

```
terraform apply --target google_project_service.project
```

実行の承認を求められるので、 yes を入力する（以降の terraform apply の実行でも同様にすること）。

（8）Cloud DNSマネージドゾーンの作成およびドメイン解決の委譲

以下のコマンドで Cloud DNS マネージドゾーンを作成する。

```
terraform apply --target google_dns_managed_zone.zone
```

Google Cloud コンソール上で、作成されたリソースを確認することができる。 マネージドゾーン名を取得し、以下のコマンドを実行してNSレコードを取得する。

```
Google Cloudloud dns record-sets list --zone <マネージドゾーン名> --
format='value(nameServers)' --flatten 'nameServers'
```

出力されたNSレコードを、ドメインのレジストラで、ドメインのネームサーバーとして設定する。設定方法は各レジストラによって異なるため、レジストラのドキュメントを参照すること。

（9）環境設定ファイルの作成

terraform.tfvars.exampleをコピーする。

```
cp terraform.tfvars.example terraform.tfvars
```

次に、コメントを参考に、これまで構築してきたGoogle Cloud・MongoDB・Auth0などの情報をterraform.tfvarsに設定する。

（10）Terraformの実行

再度、すべてのリソースを作成するために以下のコマンドを実行する。

```
terraform apply
```

実行が成功すると、以下のような出力が表示される。

```
Apply complete! Resources: * added, * changed, * destroyed.

Outputs:

plateauview_cms_url = "*"
plateauview_cms_webhook_secret = <sensitive>
plateauview_cms_webhook_url = "*"
plateauview_editor_url = "*"
plateauview_flow_url = "*"
plateauview_flow_token = <sensitive>
plateauview_geo_url = "*"
plateauview_setup_token = <sensitive>
plateauview_sdk_token = <sensitive>
plateauview_sidebar_token = <sensitive>
plateauview_sidecar_url = "*"
plateauview_tiles_url = "*"
```

これらの出力は、あとでログインする時に使用する。なお、もう一度表示したいときは terraform output コマンドで表示できる。また、sensitiveと表示されているものは、マスクされており、以下のようなコマンドで実際の値を確認すること。

変数	説明
plateauview_cms_url	PLATEAU CMSのURL。
plateauview_cms_webhook_secret	後述の「CMS インテグレーション設定」で使用。
plateauview_cms_webhook_url	後述の「CMS インテグレーション設定」で使用。
plateauview_geo_url	タイルなどを変換・処理するサーバーのURL。
plateauview_editor_url	PLATEAU EditorのURL。
plateauview_flow_url	PLATEAU FlowのURL。
plateauview_flow_token	PLATEAU Flowのトリガー作成時にで使用。
plateauview_sdk_token	PLATEAU SDK for Unity/Unreal用のトークン。SDKのUIで設定する。
plateauview_setup_token	後述のPLATEAU CMSのセットアップで使用。
plateauview_sidebar_token	ビューワのサイドバー用のAPIトークン。エディタ上でサイドバーウィジェットの設定から設定する。
plateauview_sidecar_url	サイドカーサーバーのURL。エディタ上でサイドバーウィジェットの設定から設定する。
plateauview_tiles_url	タイル配信サーバーのURL。

（11）DNS・ロードバランサ・証明書のデプロイ完了の確認

実際にcurlコマンドなどでリクエストを送って、デプロイが完了していることを確認する。

```
curl https://api.${DOMAIN}/ping
```

先ほど作成したAuth0テナントにユーザーを作成する。 その後に、届くメールでメールアドレスを認証するか、メールアドレス認証のステータスをアカウント詳細画面からVerifiedにする。必ず上記ステップでデプロイが完了していることを確認してから、Auth0のユーザーを作成する。先に作成した場合、正常にEditorやCMSにログインできなくなる。

（12）CMSインテグレーション設定

Terraformのの plateauview_cms_url のURL（https://reearth.${DOMAIN}）からPLATEAU CMSにログインする。ログイン後、ワークスペース・Myインテグレーションを作成する。

次に、インテグレーション内に以下のとおり Webhook を作成する。作成後、有効化を忘れないこと。

- URL: terraform outputsのplateauview_cms_webhook_url
- シークレット: terraform outputsのplateauview_cms_webhook_secret
- イベント: 全てのチェックボックスにチェックを入れる。

作成後、作成したワークスペースに作成したインテグレーションを追加し、オーナー権限に変更する。

先ほど作成したインテグレーションの詳細画面でインテグレーショントークンをコピーし、以下の ${REEARTH_PLATEAUVIEW_CMS_TOKEN} に貼り付けて以下のコマンドを実行する。

```
echo -n "${REEARTH_PLATEAUVIEW_CMS_TOKEN}" | Google Cloudloud secrets versions add
reearth-cms-REEARTH_PLATEAUVIEW_CMS_TOKEN --data-file=-
```

環境変数の変更を適用するため、もう一度 Cloud Run をデプロイする。

```
Google Cloudloud run deploy plateauview-api
  --region asia-northeast1
  --platform managed
  --quiet
```

（13）完了

以下のアプリケーションにログインし、正常に使用できることを確認する。ここの ${DOMAIN} はドメインである。

- Editor: terraform outputs の plateauview_editor_url の値（https:/editor.${DOMAIN}）
- CMS: terraform outputs の plateauview_cms_url の値（https://cms.${DOMAIN}）
- Flow: terraform outputs の plateauview_flow_url の値（https://flow.${DOMAIN}）

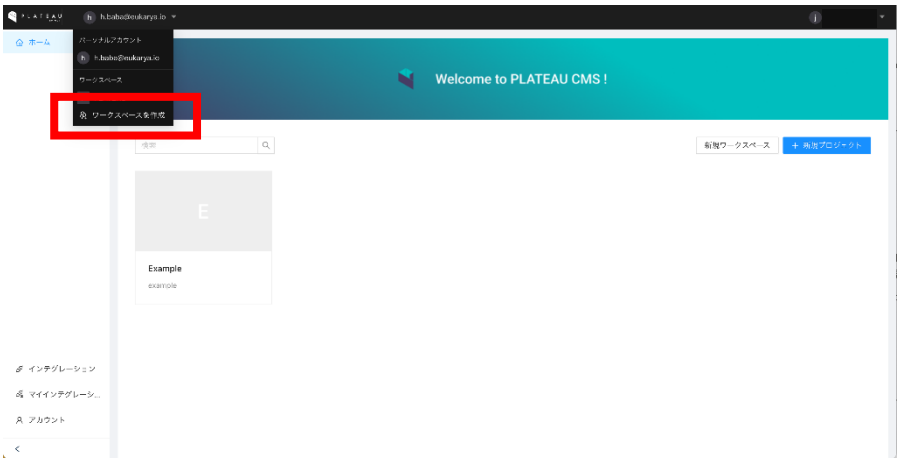
サーバー構成は以上で完了である。

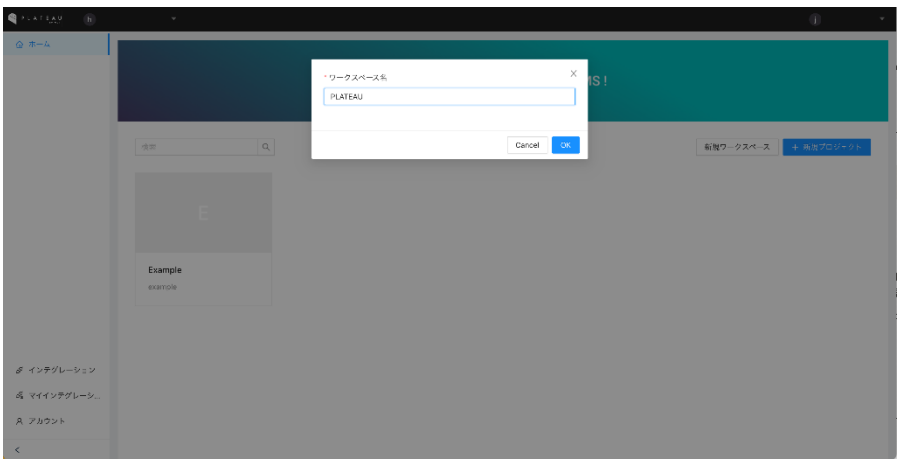
3.1.4 PLATEAU CMSの動作確認・セットアップ

PLATEAU CMS内でのプロジェクトの初期設定及びインテグレーションの設定に移る。

（1）ワークスペースの作成

- 個人ワークスペースでは他のユーザーを招待できないため、新たにワークスペースを作成する。
- PLATEAU CMSにログインし、ホーム画面に遷移
- 新規ワークスペース作成ボタンを押下
  - ヘッダー > ワークスペースを作成
  - ホーム > 新規ワークスペース
  - ワークスペース名を入力



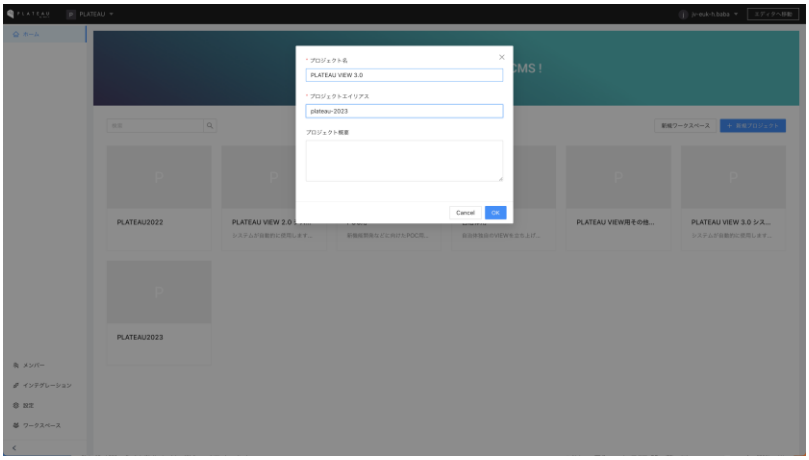


（2）プロジェクトの作成

2つのプロジェクトを作成する。

1. PLATEAU関連データセットを管理するためのプロジェクト（プロジェクトエイリアス：plateau-2024）
2. VIEWの情報を保存するためのプロジェクト（プロジェクトエイリアス：plateauview）
3. 全体設定などを保持するシステム用プロジェクト（プロジェクトエイリアス：system）

なお、プロジェクトのエイリアスとは、PLATEAU CMSから公開されるAPIの別名で、公開APIのURLの中で使われる。必須項目であり、5文字以上32文字以下の半角英数字と一部記号が使用可能（_又は-）。他プロジェクトが同じエイリアスを使うとエラーとなる。



（3）モデルの作成

以下のようなコマンドをターミナル上で実行して、各プロジェクトのモデルを自動的に作成する。

- 以下のコマンドはMacおよびLinuxを想定している。Windowsの場合はcurlを curl.exe に置き換えて実行すること。もしエラーとなる場合は —json XXX を -H ‘Content-Type: application/json’ -X POST -d XXX に置き換えて再度実行すること。
- ${TOKEN} は、terraform outputs の plateauview_setup_token の値に置き換えること。

```
curl -h 'Auhtorization: Bearer ${TOKEN}' --json '{ "type": "system", "project": "system" }' 'https://api.${DOMAIN}/setup'
curl -h 'Auhtorization: Bearer ${TOKEN}' --json '{ "type": "view", "project": "plateauview" }' 'https://api.${DOMAIN}/setup'
curl -h 'Auhtorization: Bearer ${TOKEN}' --json '{ "type": "plateau", "project": "plateau-2024" }' 'https://api.${DOMAIN}/setup'
```

（4）ワークスペース・プロジェクトの登録

作成したワークスペースやプロジェクトをシステム用プロジェクトに登録する。
CMSにアクセスし、ワークスペースに切り替えたあと「システム用」プロジェクトを開く。その後、コンテンツ→ワークスペース（モデル）を選択肢、アイテムを新規作成する。以下のように項目を入力してアイテムを保存すること。

- ワークスペースID：現在開いているURLの workspace/xxx/project/ の xxx の部分
- データカタログ用プロジェクトID：CMSでplateau-2024プロジェクトを開いたURLの workspace/xxx/project/yyy の yyy の部分
- データカタログ用プロジェクトのエイリアス：plateau-2024
- システム用プロジェクトのエイリアス：system
- データカタログのスキーマバージョン：v3
- CMS インテグレーショントークン：terraform outputs の plateauview_setup_token の値
- バックエンドアクセストークン：terraform outputs の plateauview_sidebar_token の値
- 使用ツール：flow
- CMS URL：terraform outputs の plateauview_cms_url の値

3.1.5 PLATEAU Editorの動作確認

以下のURLにログインし、ログイン画面が表示されることを確認する。

https://editor.<ドメイン>

ログインを行い、ログイン後Editorのダッシュボード画面に遷移すれば、構築は完了である。ログインできない場合は、各種デプロイ完了後にAuth0にユーザーを作成済みであること、ユーザーのメールアドレスの認証が済んでいることを確認する。なお、ログインしてもすぐにログイン画面に戻される場合は、Auth0にユーザーは存在するが、PLATEAU側のデータベース内にユーザーを作成する処理に失敗していることが考えられる。

次にPLATEAU VIEWの構築に移る。PLATEAU VIEWの作成・公開方法については3.2を参照。

3.1.6 PLATEAU Flowの動作確認

以下のURLにログインし、ログイン画面が表示されることを確認する。

https://flow.<ドメイン>

ログインを行い、ログイン後Flowのダッシュボード画面に遷移すれば、構築は完了である。ログインできない場合は、各種デプロイ完了後にAuth0にユーザーを作成済みであること、ユーザーのメールアドレスの認証が済んでいることを確認する。なお、ログインしてもすぐにログイン画面に戻される場合は、Auth0にユーザーは存在するが、PLATEAU側のデータベース内にユーザーを作成する処理に失敗していることが考えられる。

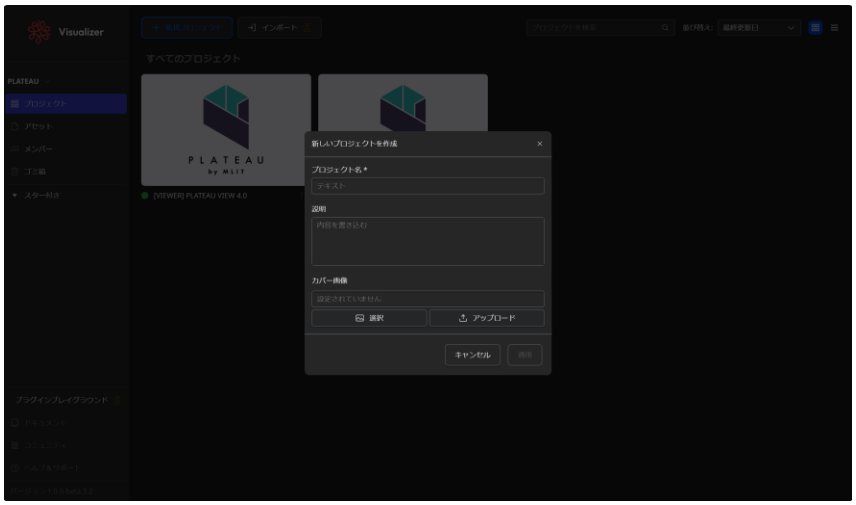
次に、2.3.2を参照して、FlowにてCMSやVisualizerと同じワークスペースにて、ワークフローとトリガーをセットアップを行い、トリガーIDを得ること。
なお、ワークフローは、以下のURLから必要な workflows.json を取得し、Flowにインポートすること。なお、品質検査/データ変換および地物型ごとにワークフローのJSONファイルが分かれているため、必要な場合はそれぞれインポートして個別にデプロイとトリガーを作成する。
https://github.com/Project-PLATEAU/PLATEAU-VIEW-4.0/tree/main/flow/engine/worker/workflow/cms/plateau4
トリガーのシークレット（トークン）は、先程の terraform outputs の plateauview_flow_token の値を使用すること。
最後にCMSに移動して設定を行う。3.1.4にて作成したシステム用プロジェクトの中の「**PLATEAU地物型**」モデルを開き、該当する地物型のアイテムを編集する。編集画面にて「Flow 品質検査 v4 Trigger ID」または「Flow 変換 v4 Trigger ID」フィールドを編集して、さきほど払い出されたトリガーIDを入力し、アイテムを保存する。以上で設定が完了となる。

3.2 PLATEAU EditorとVIEWのセットアップ

3.2.1 プロジェクトの作成

PLATEAU Editor内でのプロジェクトの作成とセットアップを行う。2つのプロジェクトを作成する。

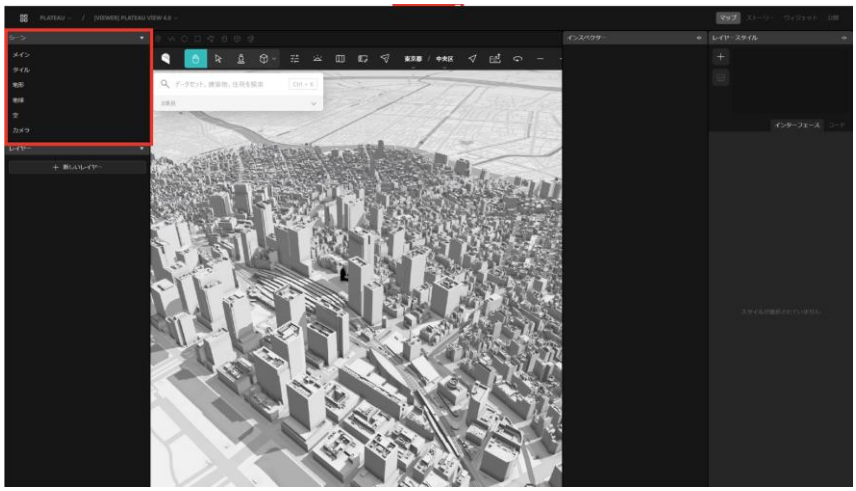
1つ目は、Editor用プロジェクト、もう一つはVIEW用プロジェクトである。以下のようにプロジェクトを新規作成する。プロジェクト名は任意である。



3.3.2 シーンの設定

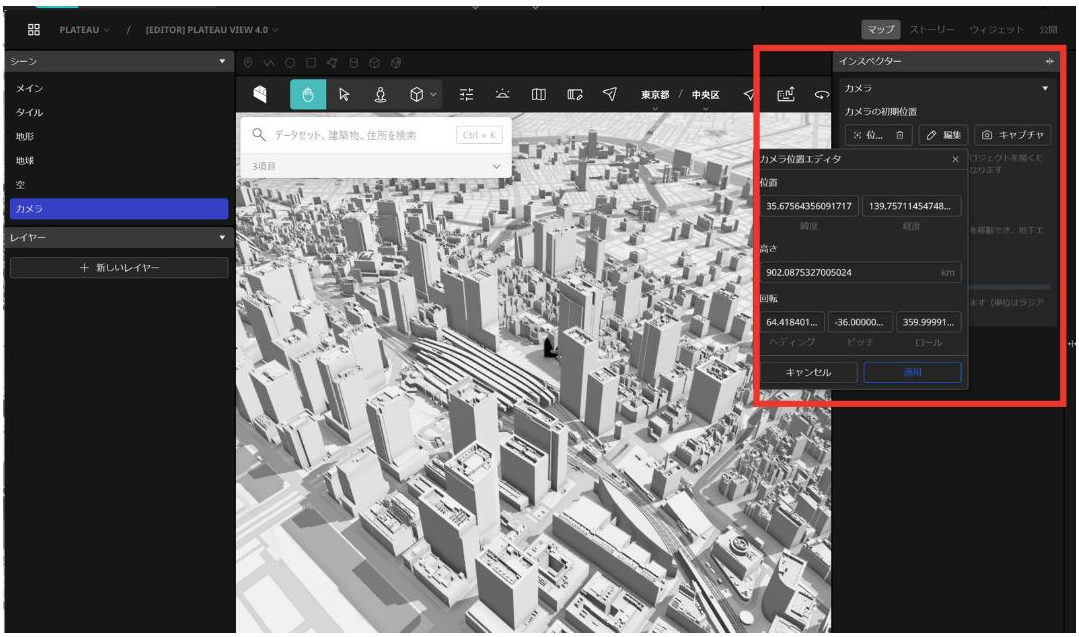
シーンでは、デジタルアースのベースマップやカメラの初期位置など、プロジェクト全体の設定等を行うことができる。またレイヤーやインフォボックスの配置などができる。

基本的な考え方として、画面左のパネル（アウトライン）または地球儀上で編集対象を選択し、画面右のパネルで設定を変更するという操作ができる。



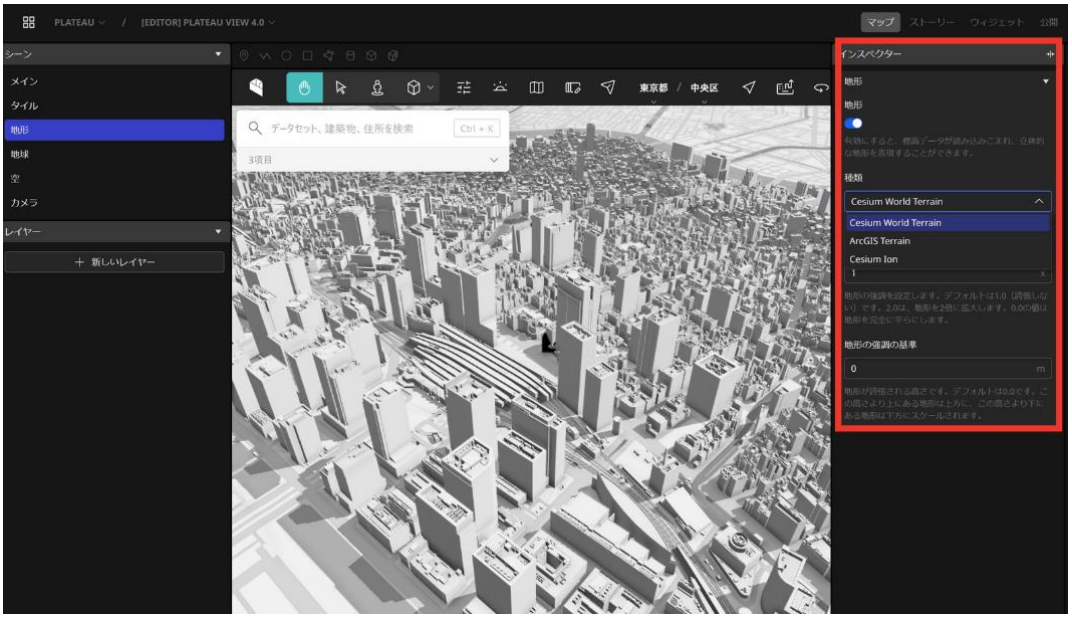
（1）初期カメラの設定

ここではPLATEAU VIEW 4.0におけるシーン全体の設定を行う。左サイドバー上部の「シーン」を選択すると、右パネルにシーンの設定が表示される。
デフォルトでは、ページロード時に北アメリカ全土が表示される設定になっている。
「カメラ初期位置の設定」では、ページロード後、最初に表示されるカメラの位置を設定できる。
「キャプチャ」を押下すると、その位置と画角がカメラ初期位置に設定される。



（2）タイル・地形の設定

タイルとは地表のベースマップの画像のことである。またTerrainとは地表の標高などに基づく三次元的形状のことである。デフォルトでは、デフォルト（Cesium）のタイルと、Cesium World Terrainを使用するようになっているため、PLATEAU独自のTerrainのURLを設定する。

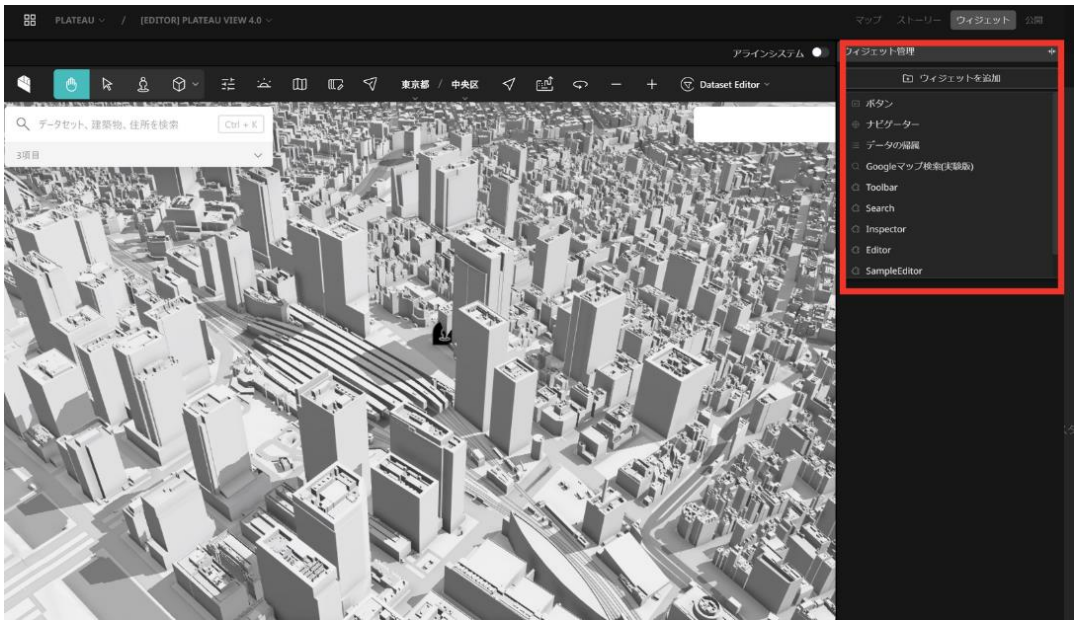


3.2.3 ウィジェットの設定

ウィジェットでは、ウインドウ上に配置される機能の設定が可能である。また、ウィジェット配置システムによって、それらのウィジェットの配置を自由に設定することができる。

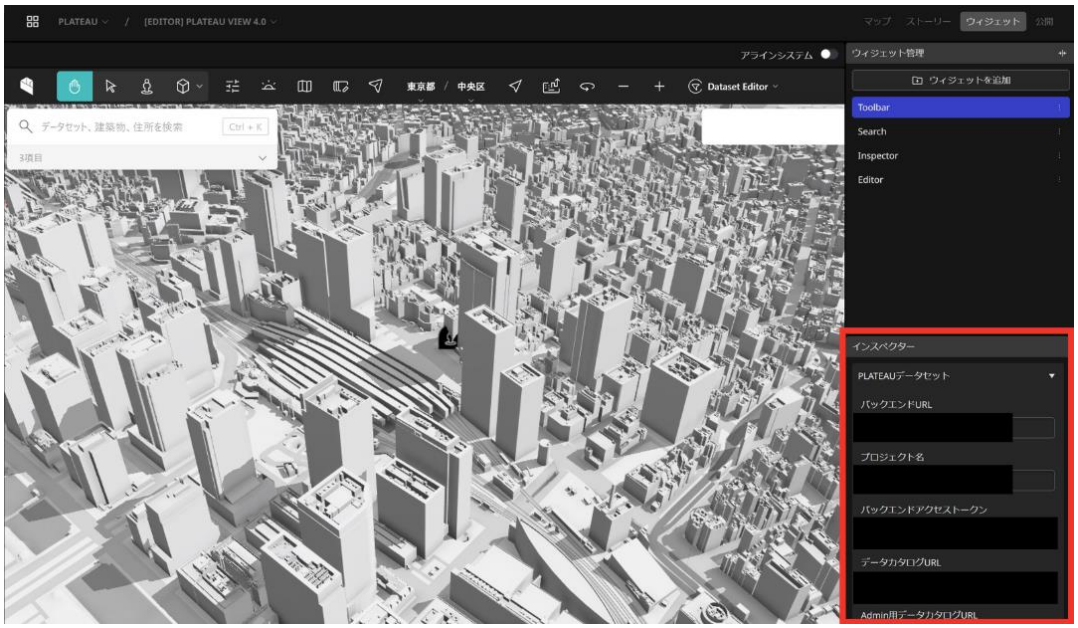
（1）ウィジェットの追加

左パネルより、ウィジェットをシーンへ追加する。



（2）ウィジェットの設定

それぞれのウィジェットは設定項目を有しており、さまざまな設定を行うことができる。左パネル上部でウィジェットを選択すると、右パネルに選択中のウィジェットの設定項目が表示される。



ウィジェット設定における詳細は以下のとおり。

- **ツールバー**
  - **バックエンド URL**: サイドカーサーバーのエンドポイント。3.1のTerraform実行結果として得た「plateauview_sidecar_url」の値を入力する。
  - **プロジェクト名**: 共有機能などのデータを保存するための、PLATEAU CMS上のプロジェクトエイリアス名。「3.1.4 PLATEAU CMSのの動作確認・セットアップ」で作成したPLATEAU CMSのPLATEAU VIEW用プロジェクトのプロジェクトエイリアスを入力する。
  - **バックエンドアクセストークン**: PLATEAU Viewサーバーの認証に使用されるアクセストークン。3.1のTerraformの実行結果として得た「plateauview_sidebar_token」の値を入力する。
  - **データカタログUR**L: データカタログ用サイドカーサーバーのエンドポイント。3.1のTerraform実行結果として得た「plateauview_sidecar_url」の値を入力する。通常、PLATEAU Backend Base URLに「datacatalog/graphql」を付与した値にとなる。
  - **Admin用データカタログUR**L: Editorで利用するデータカタログ用サイドカーサーバーのエンドポイント。3.1のTerraform実行結果として得た「plateauview_sidecar_url」の値を入力する。通常、PLATEAU Backend Base URLに「/datacatalog/admin/graphql」をつけた値になる。
  - **GeoサーバーバックエンドURL**：ジオコーディングや住所検索に利用するAPIサーバーへのエンドポイントを記載する。 3.1のTerraform実行結果として得た「plateauview_geoapi_url」の値を入力する。
  - **地理院地図タイルURL**：国土地理院が実験的に提供しているベクトルタイル形式の「地理院タイル」配信エンドポイントを入力する。
  - **Google Street View API Key**：Google Street Viewを利用するためのAPIキーを入力する。
  - **プロジェクトの公開URL**: 共有URL機能を利用した際のURL。3.1のTerraformの実行結果として得た「plateauview_reearth_url」にプロジェクトエイリアスをサブドメインとして付加したURL。
  - **フィードバックを非表示**：ご意見ご要望機能をOFFにするオプション設定。

3.3 FME Flow
3.3.1 FME Flow とその稼働環境

稼働環境の制約

CMSと連携して3D都市モデルデータの品質検査、可視化用のデータ変換は Safe Software社 (カナダ) の FME Flow (旧称 FME Server) によって行うため、同ソフトウェアとその稼働環境を用意する。

必要な FME ソフトウェアおよびその稼働環境は次のとおり。

- FME ソフトウェア
  - FME Flow：バージョン 2024.1.3 以降
  - CPU-Time：1式（注1）
- FME Flow の稼働環境
  - OS：次のページに記載されている FME Flow 対応 OS の範囲
    - FME 2024.x の場合：Legacy FME Technical Specifications（https://support.safe.com/hc/en-us/articles/25407724689805-Legacy-FME-Technical-Specifications）
    - FME 2025.x の場合：FME Platform Technical Specifications（https://support.safe.com/s/article/FME-Platform-Technical-Specifications）
  - 推奨スペック：RAM64GB以上, ディスク容量500GB以上（注: 必要なメモリ、ディスク容量は、品質検査やデータ変換の対象とするデータのサイズや同時処理数に応じて異なり、このスペックであればどんな条件でも対応できるということを保証するものではない）。
  - インターネットにアクセスできること

注1: CMSと連携した処理はいくつかのプロセスを並列で行うことがあるため、FME Flow 1ライセンス（同時実行1プロセス）のみでは対応ができない。そのため、一定のCPU時間数に達するまでの間は任意にエンジン（同時実行プロセス）数を設定できる CPU-Time を導入する必要がある。CPU-Time を利用するため、実行環境からはインターネットにアクセスできる必要がある。

Amazon Web Services S3バケット

- CMSからのリクエストに応じてFME Flow が行った品質検査や可視化用データ変換処理の結果やログファイル等は Amazon Web Services（以下「AWS」と言う）S3 経由で CMS に渡す仕組みであるため、それに必要な AWS S3 バケットをひとつ用意し、データ書き込み権限のある「アクセスキーID」及び「シークレットアクセスキー」を取得する。
- バケット名やリージョンについての制約はないので、データ書き込み権限を取得できるものであれば、既存のバケットを利用しても良い。

3.3.2 FME Flow プロジェクトファイル

FME Flow によって実行する品質検査、可視化用データ変換のフローを定義したワークスペース、CMSと連携してそれらを自動実行するための構成内容、その他の関連ファイルは、すべて次の FME Flow プロジェクトファイルに含まれている。

plateau-2024-fme-flow-project1.0.0_2025-3-26-T145449_b24627.fsproject

“-project”と拡張子 .fsproject の間の部分（バージョン、日付時刻、作成時のFMEビルド番号）は、プロジェクトファイルの更新に伴って変わることがある。

後述するように、FME Flow のウェブインターフェースの操作によってこのプロジェクトファイルを FME Flow にインポートし、いくつかの設定を行うことによってCMS 連携処理に必要なすべてのファイル、構成が再現される。

参考: 表　プロジェクトファイルに含まれるリポジトリとCMS連携処理で使用するワークスペース

リポジトリ名	CMS連携処理で使用するワークスペース
PLATEAU 2024 品質検査	• PLATEAU4 品質検査01 共通.fmw • PLTEEAU4 品質検査01-2 必須属性等.fmw • PLATEAU4 品質検査02 建築物.fmw • PLATEAU4 品質検査03 道路等の交通モデル.fmw • PLATEAU4 品質検査04 都市設備・植生.fmw • PLATEAU4 品質検査05 土地利用・都市計画決定情報・区域.fmw • PLATEAU4 品質検査06 浸水想定区域.fmw • PLATEAU4 品質検査07 土砂災害警戒区域.fmw • PLATEAU4 品質検査08 地形.fmw • PLATEAU4 品質検査09 橋梁・トンネル・地下街.fmw • PLATEAU4 品質検査10 その他の構造物 • PLATEAU4 品質検査11 水部.fmw • PLATEAU4 品質検査12 地下埋設物.fmw • PLATEAU4 品質検査13 汎用都市オブジェクトfmw
PLATEAU 2024 可視化用データ変換	• PLATEAU4 可視化用データ変化01 建築物.fmw • PLATEAU4 可視化用データ変換02 道路・鉄道・徒歩道・広場・航路.fmw • PLATEAU4 可視化用データ変換03 都市設備・植生.fmw • PLATAEU4 可視化用データ変換04 土地利用・土砂災害警戒区域.fmw • PLATEAU4 可視化用データ変換05 浸水想定区域.fmw • PLATEAU4 可視化用データ変換06 都市計画決定情報・区域.fmw • PLATEAU4 可視化用データ変換07 橋梁・トンネル・その他の構造物.fmw • PLATEAU4 可視化用データ変換08 地下街.fmw • PLATEAU4 可視化用データ変換09 地下埋設物.fmw • PLATEAU4 可視化用データ変換10 水部.fmw • PLATEAU4 可視化用データ変換11 汎用都市オブジェクト.fmw
plateau-utilities	• processing-status-notifier.fmw
plateau2024-cms	• cms-error-logger.fmw • plateau4_cms-job-submitter_conv.fmw • plateau4_cms-job-submitter_qc.fmw • plateau4_cms-request-receiver.fmw

3.3.3 環境の準備

FME Flowの構築では、Safe Software Inc.のFME Flowを稼働させるサーバー環境とFME Flowのライセンスの取得が必要となる。ここでは、FME Flow稼働のための推奨環境について解説する。

FME FlowはWindowsあるいはLinux上で動作するアプリケーションである。動作確認されているOSはWindows（11、10、Server2022、Server2019、Server2016 (FME 2024.2以前)）、Linux（Ubuntu 24.04 LTS、Ubuntu 20.04 LTS、Debian 12、Debian 11、Red Hat Enterprise Linux 9（要EPEL9リポジトリ）、 Rocky Linux 9（要EPEL9リポジトリ）、Oracle Linux 9（要 EPEL9リポジトリ））となる。最新の情報はSafe社HPのドキュメントを参照されたい。

ハードウェア要件は特に定められていないが、PLATEAU VIEWで使用する3D都市モデルの変換の実行には、64GB以上のRAMと500GB以上のディスクスペースの確保が望ましい。

3.3.4 構築の手順

以下にWindows及びLinuxへのインストール手順を示す（より詳細な設定についてはSafe社のドキュメントを参照されたい）。

（1）Windows

Safe社のFME Downloadsページからインストーラを入手し、管理者権限で実行する。実行に必要な権限はインストールするディレクトリへの書き込み権限、マシンへのLog on as a serviceの権限である。インストールはインストーラのダイアログに従って、下記のとおり実行する。

- Choose Setup Type:Expressを選択
- Destination Folder:アプリケーション、リポジトリとリソースのディレクトリを指定（デフォルトで実行）
- FME Flow Hostname:サーバーのHostnameを指定
- Web Application Server Port:80番、あるいは8080番が推奨
- Database User:インストール時にPostgreSQL データベースに作成されるFME Flow
- Databaseののユーザー名とパスワードを設定

（2）Linux

Safe社のFME DownloadsページからFME Flowのインストーラを入手し、ルートユーザー権限で ”chmod +x" コマンドによって実行可能モードを設定したうえで、ルートユーザー権限で実行する。プロンプトで指定する内容はWindowsの項を参照されたい。

3.3.5 FME Flowの設定

3.3.4でFME Flowを稼働させるサーバーを用意し、FME Flow をインストールしてライセンシング（ライセンスをインストールするための操作）完了後、FME Flow の稼働環境が整う。

稼働環境が整ったのち、FME Flow ウェブインターフェースに管理者（admin）権限のユーザーでログインし、以下の手順でプロジェクトファイルのインポートおよび各種設定を行う。

※FME Flow のウェブインターフェースの具体的な操作方法については、使用するバージョンの FME Flow 公式ドキュメントを参照すること。

※以下の説明において「メニュー」とは、FME Flow のウェブインターフェース上のメインメニューを指す。

（1）プロジェクトファイルのインポート

メニュー：Projects > Manage Projects を選択して Projects 画面に移動する。

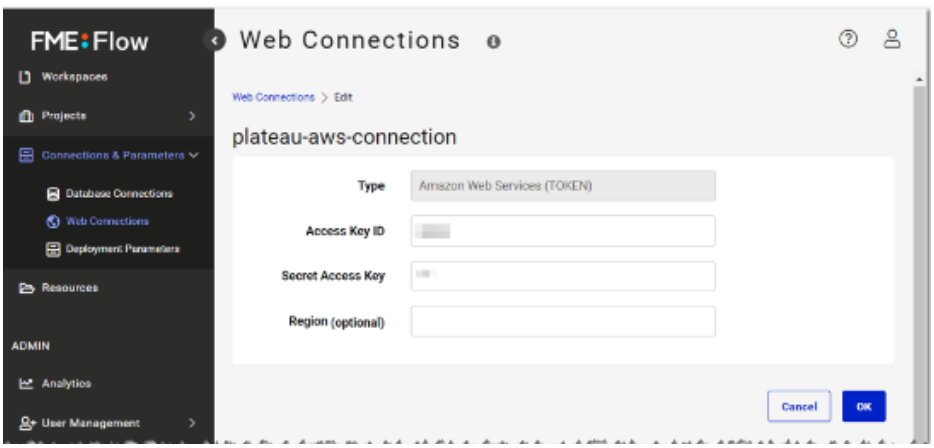
Projects 画面上部の [Import] ボタンを押下して Import Project 画面に移動し、前述のプロジェクトファイル（*.fsproject）を選択してアップロードする。

プロジェクトが展開、設定されるまでそのまま待機する（数分かかることがある）。

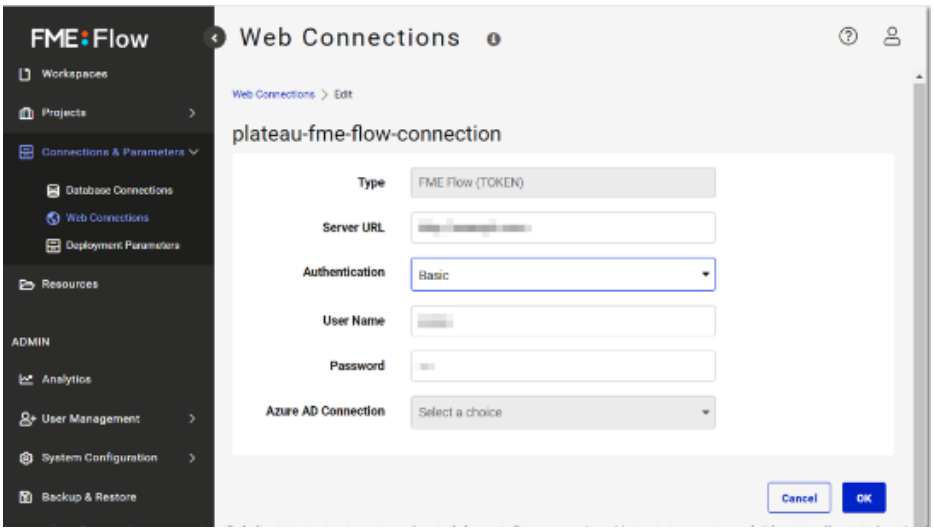
（2）接続設定（Web Connections）

メニュー：Connections & Parameters > Web Connections を選択して Web Connections 画面に移動し、次のふたつの接続パラメーターを使用環境にあわせて設定する。（接続名を押下するとパラメーター設定画面が開くので、画面上に表示される各フィールドに入力する）

- plateau-aws-connection：データ変換結果等の格納用として使用する AWS S3 バケットに接続するのに必要なパラメーター（AWS Acces Key ID, Secret Access Key）を設定する。バケット名は後述の自動処理の設定において指定する。



- plateau-fme-flow-connection：データ変換処理の一部では FME Flow が自分自身に接続して処理をリクエストするものがあり、それに必要なパラメーター（Server URL, User Name, Password）を設定する。



（3）エンジン設定（Engine Management）

メニュー：Engine Management 以下の各メニューで、エンジン数（同時実行プロセス数）の設定やキューへのエンジンの割当などを行う。以下は簡易な設定の例であり、FME Flow の使い方に習熟したら、FME Flow の公式ドキュメントに従ってさらに高度で効率的な設定を行っても良い。

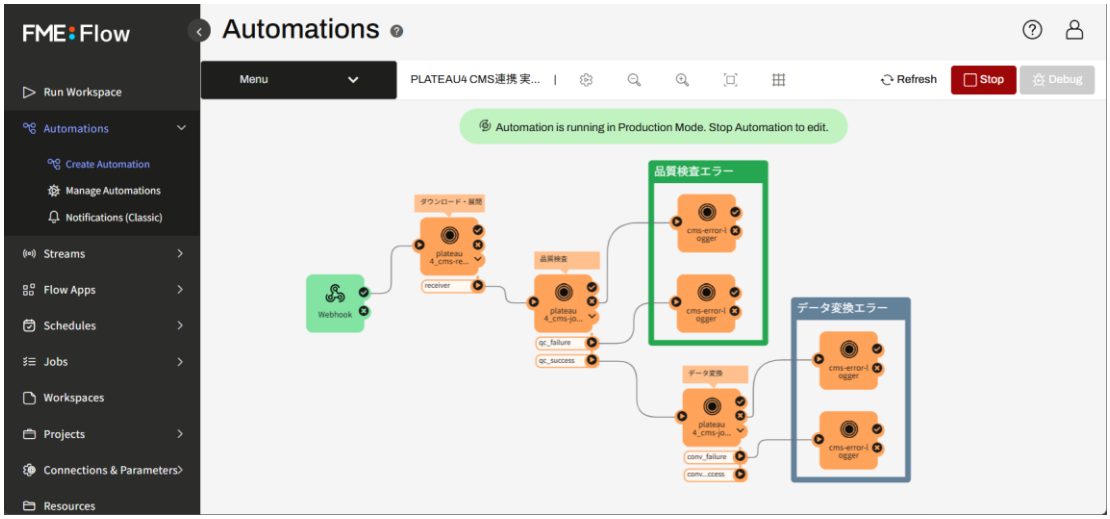
- Engines：エンジン数（同時実行可能なプロセス数）を設定する。Standard Engines（上限：FME Flow ライセンス数）と Dynamic Engines（上限なし）のエンジン数の合計は、次に掲げるキューの数 x 2以上とするのが望ましい。
- Queues：次の5つのキューを作成する。a ～ d のキューの名称は任意で良い。
    - CMSからのリクエスト受付用（対応するリポジトリ：plateau2024-cms）
    - 品質検査用（同：PLATEAU 2024 品質検査）
    - データ変換用（同：PLATEAU 2024 可視化用データ変換）
    - ユーティリティ用（同：plateau-utilities）
    - データ変換における並列処理専用（キューの名称：Parallel Processing）
- Job Routing Rules：キュー a ～ d に対応するリポジトリを割り当てる。各ルールの名称は任意で良い。
- Engine Assignment Rules：キュー a ～ e にエンジンを割り当てる。各キューの間で重複せずに、それぞれ2個以上のエンジンを割り当てるのが望ましい。

（4）自動処理の設定と起動（Automations）

メニュー：Automations > Manage Automations で Automations 画面に移動し、「PLATEAU4 CMS連携 実証環境構築用」を押下してその編集画面を開き、以下の操作、設定を行う。

- 編集画面上の左端にある Webhook トリガーを押下して詳細画面を開き、そこに表示されている Webhook URL をコピーする。このURLが、CMSからのリクエスト先となる。
- 画面上部の歯車型アイコンを押下して自動処理パラメーター編集画面（Automation Parameters Editor）を開き、S3_BUCKET_NAME パラメーターにデータ変換結果等の格納用として使用するS3バケット名を設定（初期状態で dummy と設定されているのを上書き）し、[OK] で閉じる。
- 画面右上の [Start Automation] ボタンを押下して、自動処理を起動する。

以上の設定により、Webhook トリガーから取得したURLあてにCMSから品質検査やデータ変換のリクエストをポストすると、自動処理が開始される。下図に自動処理実行中の画面例を示す。



運用上の留意事項

CMSから品質検査やデータ変換のリクエストがあると、FME Flow は対象とするデータセット（zipアーカイブ）をCMSからダウンロードし、次のリソースフォルダー以下に展開してから処理を始める。

Resources > Data > plateau2024 > downloads

データ変換が正常に終了したとき、および、毎日の自動クリーンアップ処理で取得から1週間以上経過したデータファイルは自動的に削除されるが、処理の失敗等によって削除されずに残ることもあるので、定期的に上記リソースフォルダー内をチェックし、不要なデータが残っているときは手動で削除することが望ましい。

付録　テライン配信の整備

本付録では、PLATEAUで利用する地形データについて説明する。

PLATEAU VIEWを構成するCesiumでは、quantized-mesh形式のデータをterraindb形式のファイルとして作成した「PLATEAU-Terrain」を利用している。

PLATEAU-Terrainは、国土地理院が整備した基盤地図情報 数値標高モデル5mメッシュを基本とし、5mメッシュが存在しない場所では基盤地図情報 数値標高モデル10mメッシュを利用して作成されている。terraindb形式への変換および配信には、Cesium 社が提供するサービスCesium ionを利用している。

terraindb形式のPLATEAU-Terrainは、「PLATEAU配信サービス（試験運用） チュートリアル」から無償で利用可能である（https://github.com/Project-PLATEAU/plateau-streaming-tutorial）。

Cesium ion はデータサイズが 50GB を超える場合、有料の Commercial アカウント以上を利用する必要がある。そこで、Cesium ion に依存しない Cesium 向けの地形データ変換および配信方法として、「Cesium-Terrain-Builder」と「Cesium-Terrain-Server」の活用について整理を行った。

また、terraindb形式はCesiumJS向けの地形データ形式だが、Mapbox GL JSやMapLibre GL JSなどの他の地図エンジンには対応していないという課題がある。

そこで、2024年度の事業において、CityGML形式のPLATEAU地形モデル（TIN）をMapboxやMapLibreで利用可能な地形データであるMapbox Terrain-RGBに変換するライブラリ「PLATEAU Mapbox Terrain Converter」を開発した。

以下では、「PLATEAU Mapbox Terrain Converter」を用いたMapbox Terrain-RGB形式の地形データの生成および配信についても説明する。

1.1 Cesium向けの地形データ配信

cesium-terrain-builder は、GeoTIFF形式のDEMファイルを入力データとし、Cesiumがサポートするの地形タイルを生成するコマンドラインツールである。cesium-terrain-serverはQuantized Mesh形式の地形タイルを配信するWebアプリケーションである。Quantized Mesh形式の地形タイルはCesiumで表示することが可能である。基本的にどちらもDockerを使用して動作する。

1.1.1 cesium-terrain-builderによるQuantized Meshタイルの作成

本項ではDocker Hubにあるイメージを用いた変換手法について説明する。Windowsの場合は Windows Subsystem for Linux（WSL）で作業を行うことを想定している。WSLでDockerを使用するために、Docker for Windowsをインストールし、WSLからDockerを利用できるように設定をしておく必要がある。

１．まず、以下のコマンドでDocker Hubからイメージを取得する

```
docker pull homme/cesium-terrain-builder
```

２．適当なところに入力するDEMファイル（GeoTIFF形式）を配置し、Dockerイメージを起動する。下記コマンドのpath-to-geotiffはDEMファイルが配置されているディレクトリを指定し、dockerコンテナ中の/data/ディレクトリへ割り当てている。

```
docker run -v path-to-geotiff:/data/ -t -i homme/cesium-terrain-builder /bin/bash
```

３．dockerイメージ内で以下のコマンドを実行し、タイルを生成する。

```
# mkdir /data/tiles
# ctb-tile -o /data/tiles merge.tif
```

実行すると、tilesディレクトリ以下にタイルセットが生成される。タイルセットが生成されたら、タイルディレクトリのトップに以下のような内容のlayer.jsonファイルを作成する。

”available”は各ズームレベルごとにタイルが存在する範囲を指定する。一番上がズームレベル０で、以下作成されたズームレベル全てについて昇順で記述する必要がある。

```
{
 "tilejson": "2.1.0",
 "format": "heightmap-1.0",
 "version": "1.1.0",
 "scheme": "tms",
 "tiles": ["{z}/{x}/{y}.terrain"],
 "attribution": "*attribution*",
 "projection": "EPSG:4326",
 "bounds": [-180.0, -90.0, 180.0, 90.0],
 "available": [
  [{"startX":1,"startY":0,"endX":1,"endY":0}],
  [{"startX":3,"startY":1,"endX":3,"endY":1}],
  [{"startX":7,"startY":2,"endX":7,"endY":2}],
  [{"startX":14,"startY":5,"endX":14,"endY":5}],
  [{"startX":28,"startY":11,"endX":28,"endY":11}],
  [{"startX":56,"startY":22,"endX":56,"endY":22}],
  [{"startX":113,"startY":44,"endX":113,"endY":44}],
  [{"startX":226,"startY":89,"endX":226,"endY":89}],
  [{"startX":452,"startY":178,"endX":453,"endY":178}],
  [{"startX":905,"startY":356,"endX":907,"endY":356}],
  [{"startX":1811,"startY":712,"endX":1814,"endY":713}],
  [{"startX":3623,"startY":1424,"endX":3628,"endY":1426}],
  [{"startX":7247,"startY":2848,"endX":7256,"endY":2853}],
  [{"startX":14495,"startY":5696,"endX":14512,"endY":5707}],
  [{"startX":28990,"startY":11392,"endX":29024,"endY":11415}],
  [{"startX":57981,"startY":22785,"endX":58049,"endY":22831}],
  [{"startX":115962,"startY":45571,"endX":116098,"endY":45662}]
 ]
}
```

以上でQuantized Meshタイルの準備が完了した。以降では本項で作成したQuantized Meshを配信するための手順について説明する。

1.1.2 cesium-terrain-serverのビルドおよび起動

cesium-terrain-serverはcesium-terrain-builderと同様に基本的にはDockerを用いて起動するが、Docker Hubにあるcesium-terrain-serverイメージは Docker 3.27以降のバージョンのDockerに対応していない。また、cesium-terrain-serverはGo言語で開発されているが、オリジナルのcesium-terrain-serverは現在のGo言語の仕様に合っていないため通常のビルドができない。以下の説明では、オリジナルのcesium-terrain-serverを現在の仕様に合わせて修正したバージョンでの使い方について説明する。なお、以下の説明はLinux環境下での操作手順である。Windows環境の場合はWSL2とDocker for Windowsで同様の手順を実行することができる。

1．cesium-terrain-builderで作成したタイルセットを適当なディレクトリに配置する。
以下に示す手順の説明では/home/*user-name*/terraindb/ （*user-name*はユーザ名）以下に配置しているものとする。

2．以下のコマンドで修正版cesium-terrain-serverのリポジトリをクローンする

```
git clone <https://github.com/pacificspatial/cesium-terrain-server.git>
```

3．cloneしたソースコードのトップディレクトリへ移動し、以下のコマンドを実行する

```
docker build -t "cesium-terrain-server" -f docker/Dockerfile .
```

4．ビルドが完了したら以下のコマンドを実行してサーバを立ち上げる。

```
docker run -d -p 80:8000 ¥¥
  -v /home/user-name/terraindb/:/data/tilesets/terraindb ¥¥
  cesium-terrain-server /usr/local/bin/cesium-terrain-server ¥¥
  -port 8000 -web-dir /var/www/cesium -dir /data/tilesets
```

docker -vオプションで指定するディレクトリと、cesium-terrain-server -dir オプションで指定するディレクトリ階層に注意されたい。

5．以下のコマンドで起動中のDockerコンテナに入り、/var/www/cesium/index.htmlを編集する。

```
# 起動中の Docker コンテナ名を調べる
docker container ps

# 指定したコンテナに接続
docker container exec -it <コンテナ名> bash
```

コンテナに侵入後、以下のコマンドで/var/www/cesium/index.htmlを編集する。

```
vim /var/www/cesium/index.html
```

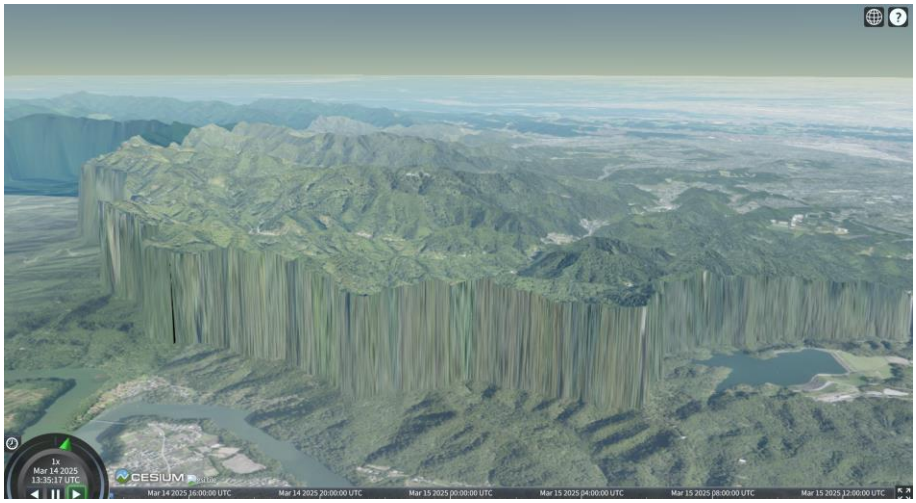
index.htmlの以下のコメント記載箇所を編集する。

```
<!DOCTYPE html>
<html lang="en">

<head>
    <!-- Use correct character set. -->
    <meta charset="utf-8">

    <!-- Tell IE to use the latest, best version (or Chrome Frame if pre-IE11). -->
    <meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1">

    <!-- Make the application on mobile take up the full browser screen and disable user scaling.
-->
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1,
minimum-scale=1, user-scalable=no">

    <title>Hello World!</title>

    <script src="/Build/Cesium/Cesium.js"></script>

    <style>
        @import url(/Build/Cesium/Widgets/widgets.css);

        html, body, #cesiumContainer {
            width: 100%;
            height: 100%;
            margin: 0;
            padding: 0;
            overflow: hidden;
        }
    </style>
</head>

<body>
    <div id="cesiumContainer"></div>
```

（次ページに続く）

```
  <script>
    // 地表の画像を変更
    // この例では地理院地図（空中写真）を指定
    var viewer = new Cesium.Viewer('cesiumContainer', {
      imageryProvider: new Cesium.createOpenStreetMapImageryProvider({
        url: '<https://cyberjapandata.gsi.go.jp/xyz/seamlessphoto/>',
        fileExtension: 'jpg',
        credit: new Cesium.Credit(
          'gsi tile', '',
          '<https://maps.gsi.go.jp/development/ichiran.html>'
        )
      }),
      baseLayerPicker: false,
      geocoder: false,
      homeButton: false
    });

    // 地形プロバイダにcesium-terrain-builderのタイルセットを指定
    var terrainProvider = new Cesium.CesiumTerrainProvider({
      url: '/tilesets/terraindb'
    });

    viewer.scene.terrainProvider = terrainProvider;

    // カメラの位置と方向を指定
    // この例では富士山付近を南側から見るように指定している
    viewer.camera.flyTo({
      destination: Cesium.Cartesian3.fromDegrees(138.728924, 35.277436, 4000.0),
      orientation: {
        heading: Cesium.Math.toRadians(0),
        pitch: Cesium.Math.toRadians(-20),
      },
    });
  </script>
</body>

</html>
```

以上の手順で問題がなければサーバを実行しているマシンのURLをブラウザで開くと、以下のように表示される。



表示に成功したらCtrl + P →Ctrl + QでDockerのシェルから離脱し、以下のコマンドで実行中のコンテナイメージを保存する。

```
docker commit <コンテナID> cesium-terrain-server:edited
```

上記のコマンドの「:edited」はタグで、元のイメージと区別するための任意の文字列を指定することができる。次回起動するときはここで保存したイメージ名＋タグ名を指定しておけば変更された状態から起動することができる。

サーバを終了するには以下のコマンドを実行する。

```
docker stop <コンテナID>
```

1.1.3

上記の手順で作成した地形データおよびサーバのデモサイトのURLを以下に示す。（デモサイトは予告なく閉鎖する場合がある。）

http://ec2-13-113-0-176.ap-northeast-1.compute.amazonaws.com/

このサイトでは次節1.2で紹介する「PLATEAU Mapbox Terrain Converter」を用いて八王子市のCityGML形式のPLATEAU地形モデル（TIN）からGeoTIFFの生成を行い、上記の手順を用いて、地形データを作成して配信を行っている。

なお、河川などの水部の高さは地形モデルには含まれていないため、0mとして表示している。地形モデルおよび水部モデルの仕様については、3D都市モデル標準製品仕様書を参照されたい。

1.2 Mapbox GL JS・MapLibre GL JS向けのテライン配信

Mapbox GL JSおよびMapLibre GL JSでの地形配信では、Mapbox-Terrain RGB（ https://docs.mapbox.com/ja/data/tilesets/reference/mapbox-terrain-rgb-v1/ ）形式のタイルを用いる。ここでは、CityGML形式のPLATEAU地形モデル（TIN）からMapbox-Terrain RGB形式のタイルセットを生成するために開発したプログラム・ライブラリ「PLATEAU Mapbox Terrain Converter」の概説および活用方法として、タイルセットの生成手順について説明を行う。

1.2.1 PLATEAU Mapbox Terrain Converter機能概要

PLATEAU Mapbox Terrain Converterの主な機能は、CityGML形式のPLATEAU地形モデル（TIN）からMapbox-Terrain RGB形式のタイルセットを生成する機能である。ライブラリと実行ファイルで構成され、ライブラリが提供する機能は以下のとおりである。

- CityGML形式のPLATEAU地形モデル（TIN）からMapbox-Terrain RGB形式のタイル生成
- 基盤地図情報標高データからMapbox-Terrain RGB形式のタイル生成
- 複数のMapbox-Terrain RGB形式のタイルセットをマージ

またオプションとして、以下の機能も提供する。

- CityGML形式のPLATEAU地形モデル（TIN）からGeoTIFF形式のDEMデータ生成

1.2.2 PLATEAU Mapbox Terrain Converterリポジトリのフォルダ構成

PLATEAU Mapbox Terrain ConverterはGitHubリポジトリ（https://github.com/Project-PLATEAU/plateau-mb-terrain-converter）で公開されている。本リポジトリのフォルダ構成は以下のとおりである。

- libplateauterrainrgb：本ライブラリのソースコード
- tools：本ライブラリを使用する実行ファイルのサンプルコード
- swig：Pythonバインディング向けSWIGインターフェース
- python_example：Pythonサンプルコード

1.2.3 動作環境

本ライブラリは以下の環境での動作を確認している。

OS:

- Windows 10
- MAC OS X
- Ubuntu 22.04

CPU、メモリ、ストレージ等: CPUに関しては特に制限はないが、メモリに関しては入力ファイルのサイズによって必要な容量が決定する。 ストレージに関しては入力ファイルの容量に加えて出力されるタイルセットの容量が必要になるが、これは入力ファイルがカバーしている範囲に依存する。ネットワーク接続は必要ない。

1.2.4 事前準備

本ライブラリは以下のライブラリに依存する。利用するにあたっては、事前にインストールしてアクセス可能な状態（PATH環境変数を設定）にしておく必要がある。

- GDAL ： 3.4.1以上
- libpng ： 1.2以上
- SQLite ： 3.3.7以上
- libxml2 ： 2.9以上
- Python 3.7 ： 3.7以上

Windows環境の場合はOSGeo4W（ https://trac.osgeo.org/osgeo4w/ ）からインストールする方法が簡便である。またMacOS環境の場合はHomebrew（ https://brew.sh/ja/ ）からインストールする方法が利用可能である。Linux環境の場合は各ディストリビューションで用いられているパッケージマネージャからインストールを行う必要がある。

また、本ライブラリをビルドする場合は以下のソフトウェアが必要である。

- C++コンパイラ ： C++17（ISO/IEC 14882:2017）に対応しているもの
- CMake
- SWIG ： Pythonバインディングの作成に必要

1.2.5 ビルドとインストール

以下に、CMakeコマンドを用いたビルド手順を示す。

```
$ mkdir build # (Windowsの場合は md build)
$ cd build
$ cmake ..
$ cmake --build . --config Release
$ cmake --install .
```

CMakeコマンドの詳細についてはCMakeドキュメント
（ https://cmake.org/cmake/help/latest/manual/cmake.1.html ）を参照されたい。

1.2.6 API説明

本項ではPLATEAU Mapbox Terrain Converterのライブラリであるlibplateauterrainrgbが提供するAPIについて説明する。

createPlateauTileset

```
namespace pmtc
{
   bool PMTC_DLL createPlateauTileset(
      const std::string &strInputTerrainCityGML,
      const std::string &strOutputTileDirectory,
      const int nMinZoomLevel,
      const int nMaxZoomLevel,
      const bool bOverwrite,
      const std::function<void(MESSAGE_STATUS, const std::string&)> &fnMessageFeedback
= nullptr,
      const std::function<void(int)> &fnProgressFeedback = nullptr
   );
}
```

CityGML形式のPLATEAU地形モデル（TIN）からMapbox-Terrain RGB形式のタイルセットを作成する。 出力ディレクトリにすでにタイルが存在している場合はbOverwrite引数によって以下のように追記される

- true　：すでに存在しているタイルに対して入力ファイルがカバーする範囲を上書きする。
- false ：すでに存在しているタイルのNoDataの部分に対して入力ファイルがカバーする範囲を追記する。

CityGML形式のPLATEAU地形モデル（TIN）からMapbox-Terrain RGB形式のタイルセットを作成する。 出力ディレクトリにすでにタイルが存在している場合はbOverwrite引数によって以下のように追記される。

ここでNoDataは出力タイル画像のうちアルファチャンネルの値が0のピクセルを指す。

引数

パラメータ名	説明
strInputTerrainCityGML	入力ファイル名（gml）
strOutputTileDirectory	出力ディレクトリ
nMinZoomLevel	最小ズームレベル
nMaxZoomLevel	最大ズームレベル
bOverwrite	上書きモード
fnMessageFeedback	メッセージコールバック関数
fnProgressFeedback	進捗コールバック関数

戻り値

パラメータ名	説明
true	変換成功
false	変換失敗

createGsiTileset

```
namespace pmtc
{
   bool PMTC_DLL createGsiTileset(
      const std::string &strInputGsiGml,
      const std::string &strOutputTileDirectory,
      const int nMinZoomLevel,
      const int nMaxZoomLevel,
      const bool bOverwrite,
      const std::function<void(MESSAGE_STATUS, const std::string&)> &fnMessageFeedback
= nullptr,
      const std::function<void(int)> &fnProgressFeedback = nullptr
   );
}
```

基盤地図情報標高データ（.xml）ファイルからMapbox-Terrain RGB形式のタイルセットを作成する。それ以外の動作はcreatePlateauTileset()関数と同様である。 標高データ中の「データなし」（-9999.0）の点はNoDataとして取り扱われる。

※本関数の動作確認は基盤地図情報標高10mメッシュ-地形図の等高線（10B）で実施している。それ以外の標高データに関しては一部未検証である。

引数

パラメータ名	説明
strInputGsiGml	入力ファイル名（gml）
strOutputTileDirectory	出力ディレクトリ
nMinZoomLevel	最小ズームレベル
nMaxZoomLevel	最大ズームレベル
bOverwrite	上書きモード
fnMessageFeedback	メッセージコールバック関数
fnProgressFeedback	進捗コールバック関数

戻り値

パラメータ名	説明
true	変換成功
false	変換失敗

mergeTilesets

```
namespace pmtc
{
   void PMTC_DLL mergeTilesets(
      const std::vector<std::string> &vstrInputDirs,
      const std::string& strOutDir,
      const bool bOverwrite,
      const std::function<void(MESSAGE_STATUS, const std::string&)> &fnMessageFeedback
= nullptr,
      const std::function<void(int)> &fnProgressFeedback = nullptr
   );
}
```

複数のタイルセットをマージする。重複するタイルが存在する場合は引数bOverwriteによって以下のようにマージされる。

- true　：引数vstrInputDirsで指定されたディレクトリの後ろにあるものが優先される。
- false　：引数vstrInputDirsで指定されたディレクトリの前にあるものが優先される。

引数

パラメータ名	説明
vstrInputDirs	入力ディレクトリ群
strOutDir	出力ディレクトリ
bOverwrite	上書きモード
fnMessageFeedback	メッセージコールバック関数
fnProgressFeedback	進捗コールバック関数

fill_zero

```
namespace pmtc
{
   void PMTC_DLL fill_zero(
      const std::string &strTileDir,
      const std::function<void(MESSAGE_STATUS, const std::string&)> &fnMessageFeedback
= nullptr,
      const std::function<void(int)> &fnProgressFeedback = nullptr
   );
}
```

指定したディレクトリ以下にあるタイル画像のNoDataのピクセルを0m
（RGBA=[1,134,160,255]）に置換する。置換は上書きで行われる。

引数

パラメータ名	説明
strTileDir	入力ディレクトリ
fnMessageFeedback	メッセージコールバック関数
fnProgressFeedback	進捗コールバック関数

terrain2gtif

```
namespace pmtc
{
   bool PMTC_DLL terrain2gtif(
      const std::string &strInputTerrainCityGML,
      const std::string &strOutputGTif,
      double dResolutionLon,
      double dResolutionLat,
      const std::function<void(MESSAGE_STATUS, const std::string &)> &fnMessageFeedback
= nullptr,
      const std::function<void(int)> &fnProgressFeedback = nullptr
   );
}
```

CityGML形式のPLATEAU地形モデル（TIN）をGeoTIFFに変換する。出力GeoTIFFのNoDataの値には-9999を割り当る。

引数

パラメータ名	説明
strInputTerrainCityGML	入力ファイル名（gml）
strOutputGTif	出力ファイル名(tif)
dResolutionLon	経度の地上解像度（°）
dResolutionLat	緯度の地上解像度（°）
fnMessageFeedback	メッセージコールバック関数
fnProgressFeedback	進捗コールバック関数

1.2.7 コマンドラインツール説明

本項ではlibplateauterrainrgbライブラリを利用して変換を行う実行ファイルの利用方法について説明する。以下に示すコマンドラインツールのソースコードは、具体的にlibplateauterrainrgbライブラリを利用する例として参照されたい。

ConvertTerrainToMapboxRGB

コマンドライン

```
convertTerrainToMapboxRGB [(options)] [input file path (CityGML)] [output directory]
```

CityGML形式のPLATEAU地形モデル（TIN）からMapbox-Terrain RGB形式のタイルセットを生成する。

引数

- 入力ファイルへのパス：入力ファイルはCityGML形式のPLATEAU地形モデル（TIN）である必要がある。
- 出力ディレクトリ：Mapbox-Terrain RGB形式のタイルセットを出力するディレクトリを指定する。出力ディレクトリにすでにタイルが存在している場合は追記される。追記の動作は下記の--overwriteオプションによる

オプション

- -min_zoom：最小ズームレベルを指定する。省略すると最小ズームレベルは６となる。
- -max_zoom：最大ズームレベルを指定する。省略すると最大ズームレベルは１５となる。
- -overwrite ：出力ディレクトリにタイルセットがすでに存在している場合にこのオプションを指定すると、すでに存在しているタイルに対して入力ファイルがカバーする範囲を上書きする。このオプションを指定しない場合は、すでに存在しているタイルのNoDataの部分に対して入力ファイルがカバーする範囲を追記する。
- -logfile [ファイル名] ：指定したファイルにログを出力する。すでに存在しているファイルを指定した場合は末尾に追記する。

ここでいうNoDataは出力タイル画像のうち、アルファチャンネルの値が0のピクセルを指す。

本コマンドおよびライブラリの変換処理は出力先に既存のタイルディレクトリを指定すると追記される。すなわち、出力先に出力対象となるタイル画像がすでに存在している場合はその画像にピクセル値を書き込み、存在しない場合は新たにタイル画像を生成する。したがって以下のコマンドで複数のファイルから一つのタイルセットを生成することができる。（下記コマンドの*path-to-output*のところに出力ディレクトリを指定）

```
for I in *.gml; do convertTerrainToMapboxRGB $I path-to-output; done
```

ConvertTerrainToMapboxRGB

コマンドライン

```
convertGsiDemToMapboxRGB [(options)] [input file path(.xml)] [output directory]
```

基盤地図情報標高ファイル（GML）ファイルからMapbox-Terrain RGB形式のタイルセットを生成する。

※本プログラムの動作確認は基盤地図情報標高10mメッシュ-地形図の等高線（10B）で実施している。それ以外の標高データに関しては一部未検証である。

引数

- 入力ファイルへのパス：- 入力ファイルへのパスを指定する。入力ファイルは基盤地図情報GML形式である必要がある。
- 出力ディレクトリ：Mapbox-Terrain RGB形式のタイルセットを出力するディレクトリを指定する。出力ディレクトリにすでにタイルが存在している場合は追記される。追記の動作は下記の--overwriteオプションによる

オプション

- -min_zoom：最小ズームレベルを指定する。省略すると最小ズームレベルは6となる。
- -max_zoom：最大ズームレベルを指定する。省略すると最大ズームレベルは１５となる。
- -overwrite ：出力ディレクトリにタイルセットがすでに存在している場合にこのオプションを指定すると、すでに存在しているタイルに対して入力ファイルがカバーする範囲を上書きする。このオプションを指定しない場合は、すでに存在しているタイルのNoDataの部分に対して入力ファイルがカバーする範囲を追記する。
- -logfile [ファイル名] ：指定したファイルにログを出力する。すでに存在しているファイルを指定した場合は末尾に追記する。

merge_tilesets

コマンドライン

```
merge_tilesets [(options)] [input directory [input directory] ...]
```

引数で指定された複数のディレクトリにあるタイルセットをマージして指定したディレクトリに出力する。

引数

- 入力ディレクトリ（複数指定可）

オプション

- -outdir (required) ：出力ディレクトリ

fill_zero_tileset

コマンドライン

```
fill_zero_tileset [input directory] [(option) output directory]
```

引数で指定されたディレクトリ内のタイル画像に対して、NoData（アルファチャンネルが0のピクセル）を0m（RGBA=[1,134,160,255]）に置換する。

引数

- 対象ディレクトリ
- 出力ディレクトリ（オプション）：出力ディレクトリを省略すると対象のディレクトリの画像を上書きする。

plateau2gtif

コマンドライン

```
plateau2gtif [input file] [output file]
        [longitude resolution] [latitude resolution]
```

CityGML形式のPLATEAU地形モデル（TIN）をGeoTIFFに変換する。

引数

- 入力ファイルへのパス：入力ファイルはCityGML形式のPLATEAU地形モデル（TIN）である必要がある。
- 出力GeoTIFFファイルへのパス
- 経度の解像度（°）
- 緯度の解像度（°）

1.2.8 Python API説明

C/C++で書かれたライブラリをPythonで利用できるようにすることをPythonバインディングという。本章ではPythonバインディングにおけるAPIの説明を示す。

コールバック関数について

SWIGの仕様の関係でコールバック関数については以下のように実装する必要がある。 まず、PMTCFeedbackクラスのサブクラスを作成して２つのコールバック関数を実装する。以下に実装例を示す。このクラスのオブジェクトを作成し、各関数のfeedback引数へ渡す。

```
class myFeedback(PMTCFeedback):
    def __init__(self):
        PMTCFeedback.__init__(self)

    def messageFeedback(self, eStatus, strMessage):
        if eStatus == MESSAGE_ERROR:
            print("ERROR : " + strMessage, file=sys.stderr)
        else:
            print(strMessage)

    def progressFeedback(self, nProgress):
        sys.stdout.write(str(nProgress) + '¥¥r')
        sys.stdout.flush()
```

CreatePlateauTileset

CityGML形式のPLATEAU地形モデル（TIN）からMapbox-Terrain RGB形式のタイルセットを作成する。 出力ディレクトリにすでにタイルが存在している場合はbOverwrite引数によって以下のように追記される。

- true　：すでに存在しているタイルに対して入力ファイルがカバーする範囲を上書きする。
- false ：すでに存在しているタイルのNoDataの部分に対して入力ファイルがカバーする範囲を追記する。

ここでいうNoDataは出力タイル画像のうちアルファチャンネルの値が0のピクセルを指す。

```
def CreatePlateauTileset(
    strInputTerrainCityGML: str,
    strOutputTileDirectory: str,
    nMinZoomLevel: int,
    nMaxZoomLevel: int,
    bOverwrite: bool,
    pFeedback: myFeedback
) -> bool:
```

引数

パラメータ名	説明
strInputTerrainCityGML	入力ファイル名（gml）
strOutputTileDirectory	出力ディレクトリ
nMinZoomLevel	最小ズームレベル
nMaxZoomLevel	最大ズームレベル
bOverwrite	上書きモード
pFeedback	コールバックオブジェクト（上記参照）

戻り値

パラメータ名	説明
True	変換成功
False	変換失敗

CreateGsiTileset

基盤地図情報標高データ（.xml）ファイルからMapbox-Terrain RGB形式のタイルセットを作成する。それ以外の動作はcreatePlateauTileset()関数と同様である。 標高データ中の「データなし」（-9999.0）の点はNoDataとして取り扱われる。

※本関数の動作確認は基盤地図情報標高10mメッシュ-地形図の等高線（10B）で実施している。それ以外の標高データに関しては一部未検証の部分がある。

```
def CreateGsiTileset(
    strInputGsiGml: str,
    strOutputTileDirectory: str,
    nMinZoomLevel: int,
    nMaxZoomLevel: int,
    bOverwrite: bool,
    pFeedback: myFeedback
) -> bool:
```

引数

パラメータ名	説明
strInputTerrainCityGML	入力ファイル名（gml）
strOutputTileDirectory	出力ディレクトリ
nMinZoomLevel	最小ズームレベル
nMaxZoomLevel	最大ズームレベル
bOverwrite	上書きモード
pFeedback	コールバックオブジェクト（上記参照）

戻り値

パラメータ名	説明
True	変換成功
False	変換失敗

MergeTilesets

複数のタイルセットをマージする。重複するタイルが存在する場合は引数bOverwriteによって以下のようにマージされる。

- true　：引数vstrInputDirsで指定されたディレクトリの後ろにあるものが優先される。
- false ：引数vstrInputDirsで指定されたディレクトリの前にあるものが優先される。

```
def MergeTilesets(
    vstrInputDirs: List[str],
    strOutDir: str,
    bOverwrite: bool,
    pFeedback: myFeedback
) -> None:
```

引数

パラメータ名	説明
vstrInputDirs	入力ディレクトリ群
strOutDir	出力ディレクトリ
bOverwrite	上書きモード
pFeedback	コールバックオブジェクト（上記参照）

fill_zero

指定したディレクトリ以下にあるタイル画像のNoDataのピクセルを0m（RGBA=[1,134,160,255]）に置換する。置換は上書きで行われる。

```
def Fill_zero(
    strTileDir: str,
    pFeedback: myFeedback
) -> None:
```

引数

パラメータ名	説明
strTileDir	入力ディレクトリ
pFeedback	コールバックオブジェクト（上記参照）

Terrain2GTif

CityGML形式のPLATEAU地形モデル（TIN）をGeoTIFFに変換する。出力GeoTIFFのNoDataの値には-9999を割り当てる。

```
def Terrain2GTif(
  strInputTerrainCityGML : str,
  strOutputGTif : str,
  dResolutionLon : float,
  dResolutionLat : float,
  pFeedback : myFeedback ) -> bool:
```

引数

パラメータ名	説明
strInputTerrainCityGML	入力ファイル名（gml）
strOutputGTif	出力ファイル名（tif）
dResolutionLon	経度の地上解像度（°）
dResolutionLat	緯度の地上解像度（°）
pFeedback	コールバックオブジェクト（上記参照）

戻り値

パラメータ名	説明
true	変換成功
false	変換失敗

1.2.9 Mapbox Terrain-RGB地形データ配信URL

本ライブラリを使用して日本全土の地形データを作成し、以下に示すURLで配信を行っている。

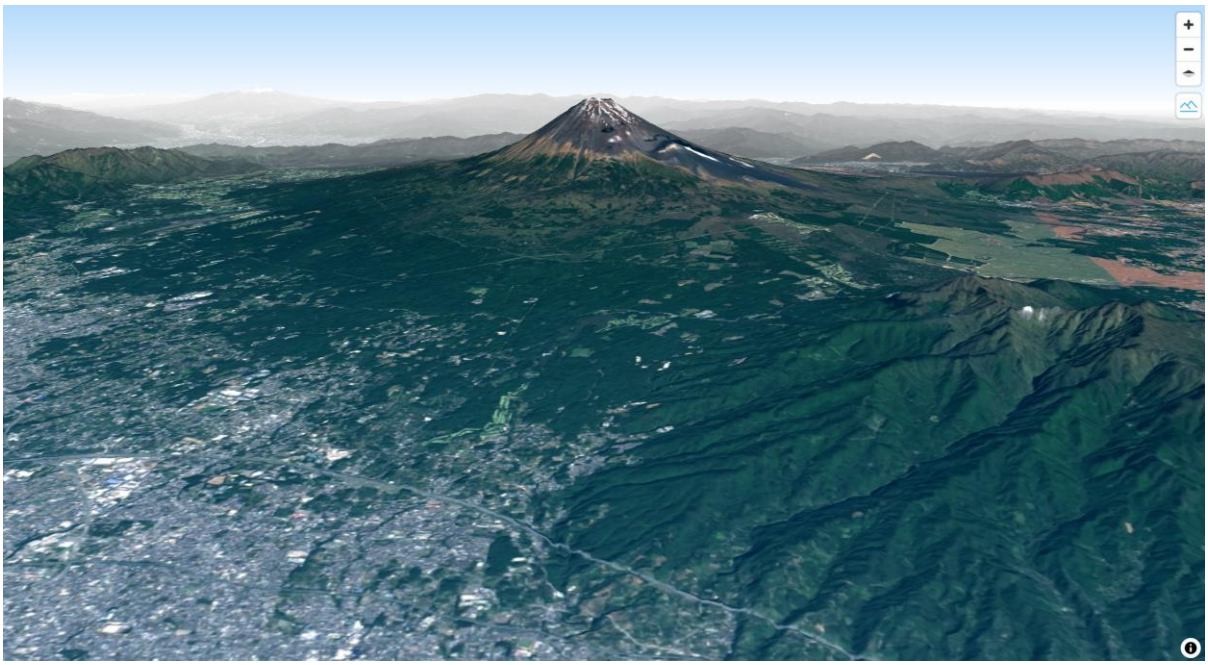
https://api.plateauview.mlit.go.jp/tiles/plateau-terrainrgb/{z}/{x}/{y}.png

ここに示した地形データは以下の手順で作成されたものである。ズームレベルは6～15に対応している。

1. CityGML形式のPLATEAU地形モデル（TIN）が2023年度までに整備されている全都市を対象に、都市ごとに変換を行い、Mapbox-Terrain RGBタイルセットを作成
2. 1で作成した全都市のMapbox-Terrain RGBタイルセットをマージ
3. CityGML形式のPLATEAU地形モデル（TIN）が無い箇所を基盤地図情報標高データ（10mメッシュ）で埋めて日本全国のMapbox-Terrain RGBタイルを作成
4. 海域等に残ったNoDataのピクセルを0mのピクセルで置換

上記の地形データをMaplibreを利用して表示したデモサイトのURLを以下に示す。（デモサイトは予告なく閉鎖する場合がある。）

http://plateau-terrainrgb-demo.s3-website-ap-northeast-1.amazonaws.com/#11.78/35.22809/138.7405/0/75



PLATEAU VIEW構築マニュアル
ver5.0
PLATEAU VIEW Setup Manual

令和7年3月　発行
国土交通省 都市局

PLATEAU
by MLIT