分野間連携データ基盤

詳細設計書

（利用者編）

第4. 0版



変更来歴

|  |  |  |  |
| --- | --- | --- | --- |
| # | 版数 | 発行年月日 | 変更内容 |
| 1 | 第1.0版 | 2020/07/07 | 新規作成 |
| 2 | - | 2020/11/30 | 来歴管理I/F（旧 原本性保証I/F）に関する記述の修正  契約確認処理の関連の記述の修正  認証認可I/Fの追加 |
| 3 | - | 2021/01/29 | 来歴管理I/Fで通信先をコンフィグから取得するように変更し、エラーコード19003Eを削除 |
| 4 | 第2.0版 | 2021/4/06 | ディレクトリ／ファイル一覧を「詳細設計書\_別紙13\_利用者コネクタ\_ディレクトリ構成」に移動 |
| 5 | 第3.0版 | 2021/11/19 | * NGSI来歴対応を反映。認証認可の機能追加。契約管理の機能追加。 * 章構成などをはじめとした詳細設計書レイアウトの調整 * 基本設計書との文言ゆれを調整 |
| 6 | 第4.0版 | 2022/6/30 | * 認証・認可に関する仕様調整の適用 * 章構成などをはじめとした詳細設計書レイアウトの調整 * 詳細設計書間での文言ゆれを調整 |

目次

[１． はじめに 1](#_Toc110853716)

[１．１ ドキュメント体系 1](#_Toc110853717)

[１．２ 適用範囲 1](#_Toc110853718)

[２． コンテナ構成 2](#_Toc110853719)

[３． 機能一覧 3](#_Toc110853720)

[４． コネクタメインサブシステム 4](#_Toc110853721)

[４．１ 内部仕様 4](#_Toc110853722)

[４．１．１ システム構成 4](#_Toc110853723)

[４．１．２ 公開インタフェース 8](#_Toc110853724)

[４．１．３ 内部データ一覧 8](#_Toc110853725)

[４．１．４ コンフィグ定義 8](#_Toc110853726)

[４．２ 機能詳細 9](#_Toc110853727)

[４．２．１ コネクタメイン機能 処理一覧 9](#_Toc110853728)

[４．２．２ 処理フロー 10](#_Toc110853729)

[４．２．３ 出力ログ 16](#_Toc110853730)

[５． 認証I/Fサブシステム 17](#_Toc110853731)

[５．１ 内部仕様 17](#_Toc110853732)

[５．１．１ システム構成 17](#_Toc110853733)

[５．１．２ 公開インタフェース 18](#_Toc110853734)

[５．１．３ コンフィグ定義 18](#_Toc110853735)

[５．２ 機能詳細 18](#_Toc110853736)

[５．２．１ 認証I/F機能 処理一覧 18](#_Toc110853737)

[５．２．２ 処理フロー 19](#_Toc110853738)

[５．２．３ 出力ログ 20](#_Toc110853739)

[６． カタログ検索I/Fサブシステム 21](#_Toc110853740)

[６．１ 内部仕様 21](#_Toc110853741)

[６．１．１ システム構成 21](#_Toc110853742)

[６．１．２ 公開インタフェース 22](#_Toc110853743)

[６．１．３ コンフィグ定義 22](#_Toc110853744)

[６．２ 機能詳細 22](#_Toc110853745)

[６．２．１ カタログ検索I/F機能 処理一覧 22](#_Toc110853746)

[６．２．２ 処理フロー 23](#_Toc110853747)

[６．２．３ 出力ログ 24](#_Toc110853748)

[７． データ交換I/Fサブシステム 25](#_Toc110853749)

[７．１ 内部仕様 25](#_Toc110853750)

[７．１．１ システム構成 25](#_Toc110853751)

[７．１．２ 公開インタフェース 26](#_Toc110853752)

[７．２ 機能詳細 26](#_Toc110853753)

[７．２．１ データ交換I/F機能 処理一覧 26](#_Toc110853754)

[７．２．２ 処理フロー 27](#_Toc110853755)

[７．２．３ 出力ログ 27](#_Toc110853756)

[８． 来歴管理I/Fサブシステム 28](#_Toc110853757)

[８．１ 内部仕様 28](#_Toc110853758)

[８．１．１ システム構成 28](#_Toc110853759)

[８．１．２ 公開インタフェース 29](#_Toc110853760)

[８．１．３ 内部データ一覧 29](#_Toc110853761)

[８．２ 機能詳細 29](#_Toc110853762)

[８．２．１ 来歴管理I/F機能 処理一覧 29](#_Toc110853763)

[８．２．２ 処理フロー 30](#_Toc110853764)

[８．２．３ 出力ログ 32](#_Toc110853765)

# はじめに

## ドキュメント体系

本書はシステムの内部設計を記載する。構成ファイル一覧を表 １‑１に記載する。

表 １‑１ 詳細設計書構成ファイル一覧

|  |  |
| --- | --- |
| # | ドキュメント名 |
|  | 詳細設計書\_共通編.docx |
|  | 詳細設計書\_提供者編.docx |
|  | 詳細設計書\_利用者編.docx |
|  | 基本設計書\_別紙2\_コンフィグパラメータ一覧.xlsx |
|  | 詳細設計書\_別紙1\_メッセージ一覧.xlsx |
|  | 詳細設計書\_別紙2\_利用者\_コネクタメイン.html |
|  | 詳細設計書\_別紙3\_利用者\_カタログ検索IF.html |
|  | 詳細設計書\_別紙4\_利用者\_データ交換IF（CADDE）.html |
|  | 詳細設計書\_別紙5\_利用者\_認証認可IF.html |
|  | 詳細設計書\_別紙6\_利用者\_来歴管理I/F.html |
|  | 詳細設計書\_別紙12\_利用者コネクタ\_ディレクトリ構成.xlsx |

## 適用範囲

本書は2022年3月版の詳細設計を対象とする。

# コンテナ構成

利用者コネクタは下図の構成で構築する。

コネクタメイン（利用者）

データ取得I/F

（CADDE）

コンテナ

コンテナ内サブシステム

HTTPS通信

凡例：

カタログ検索I/F

カタログ検索I/F

データ交換I/F

データ取得I/F

（HTTPS NGSI）

他社開発コンテナ内サブシステム

データ提供I/F

（HTTPS NGSI）

共通編記載サブシステム

データ提供I/F

（HTTPS）

データ提供I/F

（FTP）

メイン制御

来歴・履歴取得

呼び出しI/F

認証I/F

来歴管理I/F

図 ２‑１ コンテナ構成

# 機能一覧

機能一覧を記載する。

表 ３‑１詳細設計書機能一覧

| # | 機能 | 概要 | 備考 |
| --- | --- | --- | --- |
|  | コネクタメイン | * データ連携のシナリオに基づきコネクタ内の他機能を呼び出す * データ利用者に対して、外部I/F(REST API)を公開する | － |
|  | 認証認可I/F | * 認証サーバに対して、トークン確認のリクエストを送信し、データ利用者が正しいCADDEユーザかどうか認証する * 認証サーバに対して、トークン認証連携のリクエストを送信し、提供者コネクタに送信するトークンを取得する | － |
|  | カタログ検索I/F | * 横断検索サーバからカタログ情報を取得する * データ提供者のCKANからカタログ詳細情報を検索するために、提供者コネクタにカタログ詳細情報取得要求を送信する | － |
|  | データ交換I/F | * 提供者コネクタと通信(HTTPS)を確立し、データを取得する | － |
|  | データ取得I/F  （NGSI) | * データ利用者に対して、NGSIの外部I/Fを公開する * 利用者からのリクエスト情報をコネクタメインに送信する | － |
|  | 来歴管理I/F | * 来歴登録されたデータ取得時、来歴管理サービスに対して、来歴登録(受信)を行う * 限定提供データ(契約有)のデータ取得時、契約管理サービスに対して、データの受信通知を行う | ※来歴管理I/F APIとの通信方法の詳細は詳細設計にて記載  来歴管理\_機能設計書(2020年度版 1.0版).pdf[2020/09/04版]の4.来歴管理I/F APIに基づいて作成 |
|  | 来歴取得I/F | * 来歴管理サービスに対して、来歴取得のリクエストを送信し、来歴を取得する | ※来歴管理I/F APIとの通信方法の詳細は詳細設計にて記載  来歴管理\_機能設計書(2020年度版 1.0版).pdf[2020/09/04版]の4.来歴管理I/F APIに基づいて作成  ※提供者コネクタを経由しないデータ取得時に使用 |
|  | データ提供I/F  （NGSI) | * データ管理(NGSIサーバ)からNGSIにてデータを取得する |
|  | データ提供I/F  （FTP) | * データ管理(FTPサーバ)からFTPにてデータを取得する * FTPサーバのID、パスワードは、提供者コネクタ内で設定する | ※提供者コネクタを経由しないデータ取得時に使用 |
|  | データ提供I/F  (HTTPS) | * データ管理(Webサーバ)からHTTPS経由でデータを取得する * WebサーバのID、パスワードは、提供者コネクタ内で設定する | ※提供者コネクタを経由しないデータ取得時に使用 |

# コネクタメインサブシステム

## 内部仕様

### システム構成

システム構成を記載する。

利用者コネクタ Dockerコンテナ群

コネクタメイン

コンテナ

利用者のWebAPP

カタログ検索

カタログ検索

検索結果

凡例:

　　 : 対象サブシステム

　　 : その他システム

　　 : HTTPS通信

1. 認証トークン取得
2. 認証トークン検証

認証I/F

コンテナ

カタログ検索I/F

コンテナ

カタログ検索

検索結果

1. 認証トークン
2. CADDEユーザID（利用者）

図 ４．１．１‑１ カタログ検索システム構成

利用者コネクタ

Dockerコンテナ群

コネクタメインコンテナ

データ取得

（CADDE）

もしくは

データ取得

（HTTPS NGSI）

メイン

制御

利用者のWebAPP

データ交換

1. 認証トークン取得
2. 認証トークン検証
3. 認証トークン
4. CADDEユーザID  
   （利用者）

認証認可

I/F

コンテナ

ファイル

もしくは

JSON

**提供者コネクタを利用しない場合**

データ提供

I/F

（HTTPS）

データ提供

（HTTPS）

ファイル

リソース提供識別手段が

"file/http"の場合

データ提供

I/F

（FTP）

データ提供

（FTP）

ファイル

リソース提供識別手段が

"file/ftp"の場合

データ提供

I/F

(HTTPS NGSI）

データ提供

（HTTPS NGSI）

ファイル

リソース提供識別手段が

"api/ngsi"の場合

凡例:

　　 : 対象サブシステム

　　 : その他システム

　　 : HTTPS通信

　　 : 関数呼出

図 ４．１．１‑２ データ取得（CADDE） および データ取得（HTTPS NGSI）

システム構成1/2

利用者コネクタ Dockerコンテナ群

コネクタメインコンテナ

利用者のWebAPP

データ取得I/F

（CADDE）

もしくは

データ取得I/F

（HTTPS NGSI）

ファイル、

もしくは

JSON

データ交換I/F

コンテナ

データ交換

ファイルもしくはJSON

来歴管理

I/F

コンテナ

1. 受信履歴登録
2. データ証憑通知（受信）

（各完了通知）

メイン制御

**提供者コネクタを利用する場合**

※分岐以前の処理は

図 ４．１．１-２と共通

凡例:

　　 : 対象サブシステム

　　 : その他システム

　　 : HTTPS通信

ロケー

ション

サービス

ロケーション情報取得

ロケーション情報

図 ４．１．１‑３ データ取得（CADDE） および データ取得（HTTPS NGSI）

システム構成 2/2

利用者コネクタ Dockerコンテナ群

コネクタメインコンテナ

利用者のWebAPP

来歴確認

確認結果（JSON）

来歴確認

履歴検索

検索結果（JSON）

確認結果（JSON）

来歴管理

I/F

コンテナ

来歴確認

認証トークン取得

認証トークン

認証認可

I/F

コンテナ

履歴ID検索

確認結果（JSON）

来歴管理

I/F

コンテナ

来歴確認

認証トークン取得

認証トークン

認証認可

I/F

コンテナ

履歴ID検索の場合

来歴確認の場合

凡例:

　　 : 対象サブシステム

　　 : その他システム

　　 : HTTPS通信

図 ４．１．１‑４ システム構成

### 公開インタフェース

「詳細設計書\_別紙2\_利用者\_コネクタメイン.html」を参照すること。

### 内部データ一覧

以下の内部データを保持する。

表 ４．１．３‑１ 内部データ一覧

|  |  |  |
| --- | --- | --- |
| # | データ名 | 概要 |
|  | url\_file | 利用者コネクタ データ交換I/Fのデータ交換に接続するためのURL |
|  | url\_authentication | 利用者コネクタ 認証認可I/Fの認証トークン検証に接続するためのURL |
|  | url\_eventwithhash\_received | 利用者コネクタ 来歴管理I/Fの受信履歴登録に接続するためのURL |
|  | url\_history\_lineage | 利用者コネクタ 来歴管理I/Fの来歴確認に接続するためのURL |
|  | url\_history\_searchevents | 利用者コネクタ 来歴管理I/Fの履歴ID検索に接続するためのURL |

### コンフィグ定義

「基本設計書\_別紙2\_コンフィグパラメータ一覧.xlsx」を参照すること。

## 機能詳細

### コネクタメイン機能 処理一覧

表 ４．２．１‑１ インタフェース情報

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 処理名 | 概要 | URL | 関数名 | メソッド |
|  | カタログ検索 | カタログ情報を取得する。APIユーザが指定した条件に従いカタログ情報を取得する。 | /cadde/api/v4/catalog | search | GET |
|  | データ取得  (CADDE) | CADDEリクエストを受け付けメイン制御に処理を依頼する。 | /cadde/api/v4/file | files | GET |
|  | データ取得  (HTTPS NGSI) | NGSIリクエストを受け付けメイン制御に処理を依頼する。 | /cadde/api/v4/entities | retrieve\_entity | GET |
|  | 来歴確認 | 来歴管理I/Fの来歴確認を呼び出し、処理を依頼する。 | /cadde/api/v4/history/lineage/{caddec\_resource\_id\_for\_provenance} | lineage | GET |
|  | 履歴ID検索 | 来歴管理I/Fの履歴検索を呼び出し、処理を依頼する。 | /cadde/api/v4/history/searchevents | searchevents | POST |

表 ４．２．１‑２ 関数インタフェース情報

| # | 処理名 | 関数名 | 概要 | 引数 | 戻り値 |
| --- | --- | --- | --- | --- | --- |
|  | メイン制御 | fetch\_data | データ取得I/Fから呼び出され、提供者コネクタの利用有無、リソース提供手段識別子を基に処理の振り分けを行う。 | 1.リソースURL  2.リソース提供手段識別子  3.CADDEユーザID（提供者）  4.利用者トークン  5.外部IdPサーバのURL  6.NGSIオプション | [リソース提供手段識別子が"HTTPS"もしくは"FTP"の場合]  ファイルと識別情報  [リソース提供手段識別子が"NGSI"の場合]  JSON（NGSI形式）と識別情報 |

### 処理フロー

#### 独自処理

##### カタログ検索

1. HTTPクエリストリングとして、検索条件、HTTPリクエストヘッダとして、利用者トークン、外部IdPサーバのURL（以降、外部IdP-URLと称す）、検索種別、CADDEユーザID（提供者）を取得する。検索条件、CADDEユーザID（提供者）、利用者トークン、外部IdP-URLが取得できない場合は、Noneに設定する。
2. クエリストリング、検索種別、CADDEユーザID（提供者）をメッセージコードとともにログ出力する。
3. 検索種別が"meta"もしくは"detail"以外の場合、独自Exceptionを発生させる。
4. 検索条件の内容が無効な値でないかをチェックする。無効な値である場合、独自Exceptionを発生させる。
5. コンフィグファイルから以下を取得する。利用者コネクタID、利用者コネクタのシークレットがコンフィグから情報が取得できない場合、独自Exceptionを発生させる。
   * connector.json
     1. 利用者コネクタID（consumer\_connector\_id）
     2. 利用者コネクタのシークレット（consumer\_connector\_secret）
     3. ロケーションサービスのURL（location\_service\_url）
6. 利用者トークンがNoneでない場合、認証に関する処理を行う。
   1. 利用者トークン、利用者コネクタID、利用者コネクタのシークレット、外部IdP-URLを引数として、認証認可I/Fの認証トークン取得を呼び出し、レスポンスヘッダ"x-cadde-auth-token"の値を認証トークンとして取得する。認証トークン取得に失敗した場合、独自Exceptionを発生させる。
   2. 認証トークンが取得できた場合、認証トークン、利用者コネクタID、利用者コネクタのシークレットを引数として、認証認可I/Fの認証トークン検証を呼び出し、レスポンスヘッダ"x-cadde-consumer\_id"の値をCADDEユーザID（利用者）として取得する。認証トークン検証に失敗した場合、独自Exceptionを発生させる。
7. 以降、次のように処理フローを実施する。
   1. 検索種別が"meta"の場合は、横断検索処理フローを実施する。
   2. 検索種別が"detail"の場合は、詳細検索処理フローを実施する。

［横断検索処理フローの場合］

1. クエリストリングとして検索条件、検索種別、認証トークンを引数として、カタログ検索I/Fのカタログ検索を呼び出す。
2. カタログ検索I/Fのカタログ検索の応答を確認し、次のいずれかの処理を行う。
   * HTTPステータスコードが「2xx」以外の場合は、独自Exceptionを発生させる。
   * HTTPステータスコードが「2xx」の場合、検索結果に取得した検索結果とHTTPステータスコード「200」をレスポンスに設定し、応答を返す。

［提供者コネクタに対する詳細検索処理フローの場合］

1. ロケーションサービスにCADDEユーザID（利用者）を送付し、提供者コネクタのアクセスURLを取得する。取得ができない場合は、コンフィグファイルから以下を取得する。コンフィグから情報が取得できない場合、独自Exceptionを発生させる。
   * location.json
     1. コネクタロケーション（connector\_location）
        + CADDEユーザID（提供者）をキーにする
          1. 提供者コネクタのアクセスURL（provider\_connector\_url）
2. クエリストリング、検索種別、提供者コネクタのアクセスURL、認証トークンを引数として、カタログ検索I/Fのカタログ検索を呼び出す。
3. カタログ検索I/Fのカタログ検索の応答を確認し、次のいずれかの処理を行う。
   * HTTPステータスコードが「2xx」以外の場合は、独自Exceptionを発生させる。
   * HTTPステータスコードが「2xx」の場合、検索結果に取得した検索結果とHTTPステータスコード「200」をレスポンスに設定し、応答を返す。

##### データ取得（CADDE）

1. HTTPリクエストヘッダとして、利用者トークン、外部IdPサーバのURL（以降、外部IdP-URLと称す）、リソースURL、リソース提供手段識別子、CADDEユーザID（提供者）を取得する。利用者トークン、外部IdP-URL、CADDEユーザID（提供者）が設定されていない場合は、Noneを設定する。
2. リソースURL、リソース提供手段識別子、CADDEユーザID（提供者）、外部IdP-URLをメッセージコードとともにログ出力する。
3. メイン制御にパラメータを渡して処理を依頼する。

##### データ取得（HTTPS NGSI）

1. HTTPリクエストヘッダとして、利用者トークン、外部IdP-URL、リソースURL、リソース提供手段識別子、CADDEユーザID（提供者）、APIごとに利用する固有のオプションを取得する。リソースURL、リソース提供手段識別子のいずれかが取得できない場合は、独自Exceptionを発生させる。CADDEユーザID（提供者）、利用者トークン、外部IdP-URL、APIごとに利用する固有のオプションが設定されていない場合は、Noneを設定する。
2. リソース提供識別子のチェックを行う。
3. リソース提供識別子が"api/ngsi"でなければ、独自Exceptionを発生させる。
4. リソースURLの検証を行う。
5. リソースURLが以下のサポート対象のURLでない場合、独自Exceptionを発生させる。

＜サポート対象のURL＞

・https://{hostname}:{port}/v2.0/entites

・https://{hostname}:{port}/v2.0/entites/{entityId}

1. リソースURLが「https://{hostname}:{port}/v2.0/entites/」でクエリパラメータtypeが指定されていない場合、独自Exceptionを発生させる。
2. オプションヘッダをカンマ区切りの文字列からKey-Value形式のオプションに変換する。変換に失敗した場合は、独自Exceptionを発生させる。
3. リソースURL、リソース提供手段識別子、CADDEユーザID（提供者）、利用者トークン、外部IdP-URLをメッセージコードとともにログ出力する。現状Fiware-Service、Fiware-ServicePathは必須ではないためログ出力しない。
4. メイン制御にパラメータを渡して処理を依頼する。

##### 来歴確認

1. HTTPリクエストヘッダとして、利用者トークン、外部IdPサーバのURL（以降、外部IdP-URLと称す）、交換実績記録用リソースID （URL部の{caddec\_resource\_id\_for\_provenance}を該当項目として取得し、パラメータとする）、ヘッダ部の履歴取得方向、検索深度を取得する。利用者トークン、IdP-URL、交換実績記録用リソースIDが取得できない場合は、独自Exceptionを発生させる。
2. 交換実績記録用リソースID、履歴取得方向、検索深度をメッセージコードとともにログ出力する。
3. 履歴取得方向の値が"BACKWARD"、"FORWARD"、"BOTH"以外の場合は独自Exceptionを発生させる。
4. 検索深度が-1より小さい値の場合は、独自Exceptionを発生させる。
5. 利用者トークン、利用者コネクタID、利用者コネクタのシークレット、外部IdP-URLを引数として、認証I/Fの認証トークン取得を呼び出し、レスポンスヘッダ"x-cadde-auth-token"の値を認証トークンとして取得する。認証トークン取得に失敗した場合、独自Exceptionを発生させる。
6. 交換実績記録用リソースID、履歴取得方向、検索深度、認証トークンを引数として、来歴管理I/Fの来歴確認を呼び出す。交換実績記録用リソースIDは来歴管理I/Fの来歴確認用URL末尾に"/{交換実績記録用リソースID }"と追記することで設定する。
7. 来歴確認呼び出しの応答を確認し、次のいずれかの処理を行う。
   * HTTPステータスコードが「2xx」以外の場合は、独自Exceptionを発生させる。  
     また、HTTPステータスコードが「2xx」だが、レスポンスから正常に値が取得できない場合は、独自Exceptionを発生させる。
   * HTTPステータスコードが「2xx」の場合、検索結果に取得した検索結果とHTTPステータスコード「200」をレスポンスに設定し、応答を返す。

##### 履歴ID検索

1. HTTPリクエストボディとして、履歴検索用文字列、HTTPリクエストヘッダとして、利用者トークン、外部IdPサーバのURL（以降、外部IdP-URLと称す）を取得する。利用者トークン、IdP-URL、履歴検索用文字列が取得できない場合は、独自Exceptionを発生させる。
2. 履歴検索用文字列をメッセージコードとともにログ出力する。
3. 利用者トークンがNoneでない場合、利用者トークン、利用者コネクタID、利用者コネクタのシークレット、外部IdP-URLを引数として、認証I/Fの認証トークン取得を呼び出し、レスポンスヘッダ"x-cadde-auth-token"の値を認証トークンとして取得する。認証トークン取得に失敗した場合、独自Exceptionを発生させる。
4. 履歴検索用文字列、認証トークンを引数として、来歴管理I/Fの履歴ID検索を呼び出す。履歴検索用文字列はリクエストのbody部に設定する。
5. 履歴検索呼び出しの応答を確認し、次のいずれかの処理を行う。
   * HTTPステータスコードが「2xx」以外の場合は、独自Exceptionを発生させる。  
     また、HTTPステータスコードが「2xx」だが、レスポンスから正常に値が取得できない場合は、独自Exceptionを発生させる。
   * HTTPステータスコードが「2xx」の場合、検索結果に取得した検索結果とHTTPステータスコード「200」をレスポンスに設定し、応答を返す。

#### 共通処理

##### データ取得：メイン制御

1. リソース提供手段識別子が"api/ngsi"、"file/ftp"、"file/http"のいずれでもない場合は、独自Exceptionを発生させる。
2. コンフィグファイルから以下を取得する。コンフィグから情報が取得できない場合、独自Exceptionを発生させる。
   * connector.json
     1. 利用者コネクタID（consumer\_connector\_id）
     2. 利用者コネクタのシークレット（consumer\_connector\_secret）
3. 利用者トークンがNoneでない場合、利用者トークン、利用者コネクタID、利用者コネクタのシークレット、外部IdP-URLを引数として、認証I/Fの認証トークン取得を呼び出し、レスポンスヘッダ"x-cadde-auth-token"の値を認証トークンとして取得する。認証トークン取得に失敗した場合、独自Exceptionを発生させる。
4. 認証トークンが取得できた場合、認証トークン、利用者コネクタID、利用者コネクタのシークレットを引数として、認証I/Fの認証トークン検証を呼び出し、レスポンスのヘッダ"x-cadde-consumer\_id"をCADDEユーザID（利用者）として取得する。取得できない場合は、CADDEユーザID（利用者）をNoneに設定する。なお、認証トークン検証に失敗した場合、独自Exceptionを発生させる。
5. CADDEユーザID（提供者）がNoneの場合は、下記の[提供者コネクタを利用しない場合]を実施、Noneでない場合は、下記の[提供者コネクタを利用する場合]を実施。

[提供者コネクタを利用しない場合]

1. リソースURL、CADDEユーザID（利用者）、NGSIオプションをヘッダに設定して、リソース提供手段識別子に対応するコネクタ共通データ提供I/Fの処理を呼び出す。対応は以下とする。
   * リソース提供手段識別子が"file/http"の場合  
     データ提供I/F(HTTPS)
   * リソース提供手段識別子が"file/ftp"の場合  
     データ提供I/F(FTP)
   * リソース提供手段識別子が"api/ngsi"の場合  
     データ提供I/F(HTTPS NGSI)
2. データ提供I/Fの応答が成功時はステータスコード200を、ボディにサブシステムが戻す結果を設定して応答する。呼び出し先のサブシステムでExceptionが発生した場合、対応するステータスコードを設定してレスポンスを応答する。

[提供者コネクタを利用する場合]

1. コンフィグファイルから以下を取得する。コンフィグから情報が取得できない場合、独自Exceptionを発生させる。
   * connector.json
     1. ロケーションサービスのURL（location\_service\_url）
2. ロケーションサービスにCADDEユーザID（利用者）を送付し、提供者コネクタのアクセスURLを取得する。取得ができない場合は、コンフィグファイルから以下を取得する。コンフィグから情報が取得できない場合、独自Exceptionを発生させる。
   * location.json
     1. コネクタロケーション（connector\_location）
        + CADDEユーザID（提供者）をキーにする。
          1. 提供者コネクタのアクセスURL（provider\_connector\_url）
3. リソース提供手段識別子が"api/ngsi"の場合は、NGSIオプションを辞書型から"key:value,・・・"形式に変換する。変換に失敗した場合は、独自Exceptionを発生させる。リソース提供手段識別子が"api/ngsi"でない場合は、空の文字列を設定する。
4. リソースURL、リソース提供手段識別子、提供者コネクタのアクセスURL、認証トークン、NGSIオプションをヘッダに設定してデータ交換I/Fのデータ交換を呼び出す。
5. 呼び出し結果を確認し、以下を確認する。
   * ステータスコードが「200」かつ、レスポンスの以下ヘッダに値が設定されている場合は、記載の通りにデータを取得する。
     + - 1. "x-cadde-provenance"は、識別情報１としてデータを取得する。
         2. "x-cadde-provenance-management-service-url"は来歴管理サービスURLとしてデータを取得する。
         3. "x-cadde-contract-id"は、取引IDとしてデータを取得する。
         4. "x-cadde-contract-type"は、契約形態としてデータを取得する。
         5. "x-cadde-contract-management-service-url"は、契約管理サービスURLとしてデータを取得する。
   * レスポンスのHTTPステータスコードが「2xx」でない場合は、独自Exceptionを発生させる。
6. 取引ID、契約管理サービスURLに値が設定されている場合は、以下の処理を行う。
   1. 取得したデータよりハッシュアルゴリズムがSHA512のハッシュ値を生成する。
   2. CADDEユーザID（利用者) 、取引ID、ハッシュ値、契約管理サービスURL、認証トークンを引数として来歴管理I/Fのデータ証憑通知（受信）を呼び出す。来歴管理I/Fで例外が発生した場合は、該当するステータスコードとメッセージをレスポンスに設定する。
7. 呼び出し結果に識別情報１が設定されている場合は、以下の処理を行う。
   1. CADDEユーザID（利用者）を確認し、Noneの場合は、独自Exceptionを発生させる。
   2. CADDEユーザID（利用者）、識別情報１、CADDEユーザID（提供者）、来歴管理サービスURL、認証トークンを引数として、来歴管理I/Fの受信履歴登録を呼び出す。
   3. 実行結果のヘッダ"x-cadde-provenance"から識別情報2を取得する。呼び出し結果に交換実績記録用リソースIDが設定されていない（空文字が設定されている）場合は、識別情報2に空文字（""）を設定する。
8. データ交換応答のボディとHTTPステータスコード「200」をレスポンスに設定し、レスポンスを返す。レスポンス作成にはflaskライブラリを利用する。レスポンスの"attachment\_filename"はリソースURLから取得したファイル名に設定する。レスポンスのヘッダに識別情報2、レスポンスヘッダ情報を設定する。  
   レスポンスヘッダの"x-cadde-provenance"に識別情報2を設定する。

### 出力ログ

独自Exceptionに付与するエラーメッセージおよび出力ログの情報は「詳細設計書\_別紙1\_メッセージ一覧.xlsx」を参照すること。

# 

# 認証I/Fサブシステム

## 内部仕様

### システム構成

システム構成を記載する。

利用者コネクタ Dockerコンテナ群

支援サービス群

コネクタ

メイン

コンテナ

認証認可

I/F

コンテナ

認証サーバ

認証トークン取得

認証トークン

認証トークン検証

CADDEユーザID

（利用者）

認証トークン取得

認証トークン

認証トークン検証

CADDEユーザID

（利用者）

凡例:

　　 : 対象サブシステム

　　 : その他システム

　　 : HTTPS通信

図 ５．１．１‑１ システム構成

### 公開インタフェース

「詳細設計書\_別紙5\_利用者\_認証IF.html」を参照すること。

### コンフィグ定義

「基本設計書\_別紙2\_コンフィグパラメータ一覧.xlsx」を参照すること。

## 機能詳細

### 認証I/F機能 処理一覧

表 ５．２．１‑１ インタフェース情報

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 処理名 | 概要 | URL | 関数名 | メソッド |
|  | 認証トークン取得 | 認証サーバに対して、トークンフェデレーションリクエストを行い、認証トークンを返す | /token\_federation | token\_federation | GET |
|  | 認証トークン検証 | 認証サーバに対して、トークンイントロスペクションを行い、CADDEユーザID（利用者）を返す | /token\_introspect | token\_introspect | GET |

### 処理フロー

#### 認証トークン取得

1. HTTPリクエストヘッダとして、利用者コネクタID、利用者コネクタのシークレット、利用者トークン、外部IdPサーバのURL（以降、外部IdP-URLと称す）を取得する。
2. 利用者コネクタのシークレット、外部IdP-URL、利用者トークン、利用者コネクタIDをメッセージコードとともにログ出力する。
3. コンフィグファイルから以下を取得する。コンフィグから情報が取得できない場合、独自Exceptionを発生させる。
   * authentication.json
     1. 認証サーバアクセスURL（authentication\_server\_url）
4. 認証サーバにPOST通信でHTTPリクエストを発行する。リクエストのbody部に以下の値を設定する。  
   client\_id=（利用者コネクタID）&client\_secret=（利用者コネクタのシークレット）&token=（利用者トークン）& idp\_url =（外部IdP-URL）
5. 実行結果を確認し、次のいずれかの処理を行う。
   * 実行結果でエラーが発生している場合は、独自Exceptionを発生させる。エラーが発生しているかの判定は、実行結果のHTTPステータスコードが「2xx」以外の場合にエラーと判定する。
   * HTTPステータスコードが「2xx」かつレスポンスbody部の"active"の値が"true"でなかった場合、独自Exceptionを発生させる。
   * HTTPステータスコードが「2xx」かつ、レスポンスbody部の"active"の値が"true"の場合、レスポンスbody部の"auth\_token"の値を取得し、認証トークンとする。
6. レスポンスにHTTPステータスコード「200」、ヘッダ"x-cadde-auth-token"に認証トークンを設定し、応答を返す。

#### 認証トークン検証

1. HTTPリクエストヘッダとして、認証トークン、利用者コネクタID、利用者コネクタのシークレットを取得する。
2. 認証トークン、利用者コネクタID、利用者コネクタのシークレット、提供者コネクタIDをメッセージコードとともにログ出力する。
3. コンフィグファイルから以下を取得する。コンフィグから情報が取得できない場合、独自Exceptionを発生させる。
   * authentication.json
     1. 認証サーバアクセスURL（authentication\_server\_url）
4. 認証サーバにPOST通信でHTTPリクエストを発行する。リクエストのbody部に以下の値を設定する。  
   client\_id=（関連ID）&client\_secret=（関連シークレット）&token=（認証トークン）
5. 実行結果を確認し、次のいずれかの処理を行う。
   * 実行結果でエラーが発生している場合は、独自Exceptionを発生させる。エラーが発生しているかの判定は、実行結果のHTTPステータスコードが「2xx」以外の場合にエラーと判定する。
   * HTTPステータスコードが「2xx」かつレスポンスbody部の"active"の値が"true"でなかった場合、独自Exceptionを発生させる。
   * HTTPステータスコードが「2xx」かつ、レスポンスbody部の"active"の値が"true"の場合、レスポンスbody部の"username"の値を取得し、CADDEユーザID（利用者）とする。
6. レスポンスにHTTPステータスコード「200」、ヘッダ"x-cadde-consumer\_id"にCADDEユーザID（利用者）を設定し、応答を返す。

### 出力ログ

独自Exceptionに付与するエラーメッセージおよび出力ログの情報は「詳細設計書\_別紙1\_メッセージ一覧.xlsx」を参照すること。

# 

# カタログ検索I/Fサブシステム

## 内部仕様

### システム構成

システム構成を記載する。

利用者コネクタ Dockerコンテナ群

支援サービス群

コネクタ

メイン

コンテナ

カタログ検索

カタログ

検索I/F

コンテナ

横断検索

（CKAN）

カタログ横断検索

検索結果

検索結果

提供者コネクタ Dockerコンテナ群

カタログ

検索I/F

コンテナ

カタログ詳細検索

検索結果

横断検索の場合

詳細検索の場合

凡例:

　　 : 対象サブシステム

　　 : その他システム

　　 : HTTPS通信

図 ６．１．１‑１ システム構成

### 公開インタフェース

「詳細設計書\_別紙3\_利用者\_カタログ検索IF.html」を参照すること。

### コンフィグ定義

「基本設計書\_別紙2\_コンフィグパラメータ一覧.xlsx」を参照すること。

## 機能詳細

### カタログ検索I/F機能 処理一覧

表 ６．２．１‑１ インタフェース情報

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 処理名 | 概要 | URL | 関数名 | メソッド |
|  | カタログ検索 | 横断検索サーバにカタログ検索の要求を行うか、提供者コネクタに詳細カタログ検索の要求を行う。APIユーザが指定した条件に従いカタログ情報を取得する。 | /cadde/api/v4/catalog | search | GET |

### 処理フロー

#### カタログ検索

1. HTTPクエリストリングとして検索条件、HTTPリクエストヘッダとして、検索種別、提供者カタログ検索コネクタURL（横断検索の場合は不要）、認証トークンを取得する。検索種別がmetaの場合横断検索処理フロー、detailの場合、詳細検索処理フローを実施する。

[横断検索の場合]

1. クエリストリングと検索種別をメッセージコードとともにログ出力する。
2. コンフィグファイルから以下を取得する。コンフィグから情報が取得できない場合、独自Exceptionを発生させる。
   * ckan.json
     1. 横断検索サイト URL（ckan\_url）
3. クエリストリング、CKAN URLを参照し、以下の通り検索用URLを作成する。  
   検索用URL = CKAN URL + クエリストリング
4. 検索用URLから検索結果を取得する。
5. 実行結果を確認し、次のいずれかの処理を行う。
   * エラーが発生している場合は、独自Exceptionを発生させる。エラーが発生しているかの判定は、実行結果のHTTPステータスコードが「2xx」以外の場合にエラーと判定する。
   * HTTPステータスコードが「2xx」の場合は、実行結果から検索結果をHTTPステータスコード「200」をレスポンスに設定し、応答を返す。レスポンス作成にはflaskライブラリを利用する。

[詳細検索の場合]

1. HTTPリクエストヘッダの内容について以下を確認する。
   1. 提供者カタログ検索コネクタURLが取得できない場合は、独自Exceptionを発生させる。
   2. 認証トークンが取得できない場合は、独自Exceptionを発生させる。
2. クエリストリング、提供者カタログ検索コネクタURL、認証トークン、検索種別をメッセージコードとともにログ出力する。
3. クエリストリング、提供者カタログ検索コネクタURLを参照し、以下の手法で提供者コネクタへのカタログ検索用URL（以降検索URL）を作成する。  
   検索用URL = 提供者カタログ検索コネクタURL + クエリストリング
4. 検索用URL宛に、認証トークンをヘッダに設定して提供者コネクタのカタログ検索I/Fのカタログ検索を呼び出す。
5. 実行結果を確認し、次のいずれかの処理を行う。
   * エラーが発生している場合は、独自Exceptionを発生させる。エラーが発生しているかの判定は、実行結果のHTTPステータスコードが「2xx」以外の場合にエラーと判定する。
   * HTTPステータスコードが「2xx」の場合は、HTTPステータスコード「200」をレスポンスに設定し、実行結果を返す。レスポンス作成にはflaskライブラリを利用する。

### 出力ログ

独自Exceptionに付与するエラーメッセージおよび出力ログの情報は「詳細設計書\_別紙1\_メッセージ一覧.xlsx」を参照すること。

# データ交換I/Fサブシステム

## 内部仕様

### システム構成

システム構成を記載する。

提供者コネクタ Dockerコンテナ群

利用者コネクタ Dockerコンテナ群

コネクタ

メイン

コンテナ

データ交換

ファイルもしくは

JSONと

識別情報

データ交換I/F

コンテナ

データ交換I/F

コンテナ

凡例:

　　 : 対象サブシステム

　　 : その他システム

　　 : HTTPS通信

データ交換

ファイルもしくは

JSONと

識別情報

図 ７．１．１‑１ システム構成

### 公開インタフェース

「詳細設計書\_別紙4\_利用者\_データ交換IF（CADDE）.html」を参照すること。

## 機能詳細

### データ交換I/F機能 処理一覧

表 ７．２．１‑１ インタフェース情報

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 処理名 | 概要 | URL | 関数名 | メソッド |
|  | データ交換 | CADDEインタフェースを用いて対象の提供者コネクタにデータ交換の要求を送信する。 | /cadde/api/v4/file | files | GET |

### 処理フロー

#### データ交換

1. HTTPリクエストヘッダとして、リソースURL、リソース提供手段識別子、提供者データ交換コネクタURL、認証トークン、NGSIオプションを取得する。NGSIオプションが取得できない場合はNoneに設定する。認証トークンが取得できない場合は、独自Exceptionを発生させる。
2. リソースURL、リソース提供手段識別子、提供者データ交換コネクタURL、NGSIオプションをメッセージコードとともにログ出力する。
3. リソースURL、リソース提供種別識別子、認証トークン、NGSIオプションをヘッダとして、提供者データ交換コネクタURLを接続先に設定し、提供者コネクタデータ交換I/Fのデータ交換を呼び出す。
4. 実行結果を確認し、次のいずれかの処理を行う。
   * エラーが発生している場合は、独自Exceptionを発生させる。エラーが発生しているかの判定は、実行結果のHTTPステータスコードが「2xx」以外の場合にエラーと判定する。
   * HTTPステータスコードが「2xx」かつリソース提供手段識別子が"api/ngsi"の場合は、実行結果からコンテキスト情報、レスポンスヘッダ情報取得し、HTTPステータスコード「200」をレスポンスに設定して、応答を返す。レスポンス作成にはflaskライブラリを利用する。  
     なお、レスポンスヘッダに以下を設定する。値が取得できない場合は、空文字("")を設定する。
   1. 識別情報（x-cadde-provenance）
   2. 契約管理サービスURL（x-cadde-provenance-management-service-url）
   3. 取引ID（x-cadde-contract-id）
   4. 契約形態（x-cadde-contract-type）
   5. 契約管理サービスURL（x-cadde-contract-management-service-url）

* HTTPステータスコードが「2xx」かつリソース提供手段識別子が"api/ngsi"でない場合は、実行結果からファイル、レスポンスヘッダ情報取得し、HTTPステータスコード「200」をレスポンスに設定して、応答を返す。なお、レスポンスには以下を含める。
  1. レスポンスの"attachment\_filename"はリソース識別から取得したファイル名に設定する。
  2. レスポンスヘッダに以下を設定する。値が取得できない場合は、空文字("")を設定する。
     1. 識別情報（x-cadde-provenance）
     2. 契約管理サービスURL（x-cadde-provenance-management-service-url）
     3. 取引ID（x-cadde-contract-id）
     4. 契約形態（x-cadde-contract-type）
     5. 契約管理サービスURL（x-cadde-contract-management-service-url）

### 出力ログ

独自Exceptionに付与するエラーメッセージおよび出力ログの情報は「詳細設計書\_別紙1\_メッセージ一覧.xlsx」を参照すること。

# 来歴管理I/Fサブシステム

## 内部仕様

### システム構成

システム構成を記載する。

利用者コネクタ Dockerコンテナ群

来歴管理I/F

コンテナ

コネクタ

メイン

コンテナ

受信履歴登録

受信履歴登録

来歴管理

サービス

受信履歴登録

識別情報

識別情報

来歴確認

来歴確認

来歴確認

確認結果

確認結果

履歴ID検索

履歴ID検索

検索結果

検索結果

履歴ID検索

受信履歴登録の場合

来歴確認の場合

履歴ID検索の場合

凡例:

　　 : 対象サブシステム

　　 : その他システム

　　 : HTTPS通信

データ証憑通知

実行結果

履歴検索

呼び出しI/F

データ証憑通知

実行結果

データ証憑通知（受信）

図 ８．１．１‑１ システム構成

### 公開インタフェース

「詳細設計書\_別紙6\_利用者\_来歴管理I/F.html」を参照すること。

### 内部データ一覧

以下の内部データを保持する。

表 ８．１．３‑１ 内部データ一覧

|  |  |  |
| --- | --- | --- |
| # | データ名 | 概要 |
|  | url\_eventwithhash\_received | 利用者コネクタ 来歴管理I/Fの受信履歴登録に接続するためのURL |
|  | url\_history\_lineage | 利用者コネクタ 来歴管理I/Fの来歴確認に接続するためのURL |
|  | url\_history\_searchevents | 利用者コネクタ 来歴管理I/Fの履歴検索に接続するためのURL |

## 機能詳細

### 来歴管理I/F機能 処理一覧

表 ８．２．１‑１ インタフェース情報

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 処理名 | 概要 | URL | 関数名 | メソッド |
|  | データ証憑通知  （受信） | データ取得時、来歴管理にデータ証憑通知（受信）を依頼する。 | /voucher/received | voucherreceived | GET |
|  | 受信履歴登録 | データ取得時、来歴管理に受信履歴登録を行い、識別情報を取得する。 | /eventwithhash/received | received | GET |
|  | 来歴確認 | 来歴管理呼に指定された交換実績記録用リソースIDから始まる来歴確認を依頼する。 | /cadde/api/v4/history/lineage/{caddec\_resource\_id\_for\_provenance} | lineage | GET |
|  | 履歴ID検索 | ボディ部に検索条件を指定し、来歴管理に履歴検索処理を依頼する。 | /cadde/api/v4/history/searchevents | searchevents | POST |

### 処理フロー

#### データ証憑通知（受信）

1. HTTPリクエストヘッダとして、CADDEユーザID（提供者）、CADDEユーザID（利用者）、取引ID、ハッシュ値、契約管理サービスURL 、認証トークンを取得する。
2. CADDEユーザID（提供者）、CADDEユーザID（利用者） 、取引ID、ハッシュ値、契約管理サービスURLをメッセージコードとともにログ出力する。認証トークンについてはログ出力しない。
3. CADDEユーザID（利用者）、取引ID、ハッシュ値、認証トークンをヘッダとして、取得した契約管理サービスURLへHTTPリクエストを行い、契約管理サービスを呼び出す。
4. 実行結果を確認し、次のいずれかの処理を行う。

* エラーが発生している場合は、独自Exceptionを発生させる。
* 通知に成功した場合、レスポンスにHTTPステータスコード「200」を設定し、応答を返す。

#### 受信履歴登録

1. HTTPリクエストヘッダとして、CADDEユーザID（提供者）、CADDEユーザID（利用者）、識別情報（交換実績記録用リソースID）、来歴管理サービスURL、認証トークンを取得する。
2. CADDEユーザID（提供者）、CADDEユーザID（利用者）、識別情報をメッセージコードとともにログ出力する。認証トークンについてはログ出力しない。
3. 来歴管理I/Fの履歴登録APIにAPI定義から自動生成したクライアントを使用してHTTPリクエストを発行する。Body部を以下の通り設定する。  
   {  
   　　　"cdldatamodelversion": "2.0",  
   　　　"cdleventtype": "Received",  
   　　　"dataprovider": （CADDEユーザID（提供者））,  
   　　　"datauser": （CADDEユーザID（利用者））,  
   　　　"cdlpreviousevents": （識別情報）  
   }  
   また、リクエストヘッダに以下を設定する。
   * "content-type"にapplication/jsonを設定する。
   * "Authorization"に認証トークンを設定する。
4. 実行結果を確認し、次のいずれかの処理を行う。
   * エラーが発生している場合は、独自Exceptionを発生させる。
   * 取得に成功した場合、イベント識別子（cdleventid）を取得し、識別情報とする。レスポンスにHTTPステータスコード「200」、ヘッダに識別情報を設定し、応答を返す。

#### 来歴確認

1. HTTPリクエストヘッダとして、認証トークン、交換実績記録用リソースID （URL部の{caddec\_resource\_id\_for\_provenance}を該当項目として取得し、パラメータとする）、ヘッダ部の履歴取得方向、検索深度を取得する。
2. 交換実績記録用リソースID、履歴取得方向、検索深度をメッセージコードとともにログ出力する。
3. 履歴取得方向の値が"BACKWARD"、"FORWARD"、"BOTH"以外の場合は、独自Exceptionを発生させる。
4. 検索深度が-1より小さい値の場合は、独自Exceptionを発生させる。
5. コンフィグファイルから以下を取得する。コンフィグから情報が取得できない場合、独自Exceptionを発生させる。
   * provenance.json
     1. 来歴管理サービスへのアクセスのベースURL（provenance\_management\_api\_url）
6. 認証トークン、交換実績記録用リソースID、履歴取得方向、検索深度を引数として、来歴管理サービスの来歴確認を呼び出す。交換実績記録用リソースIDは来歴管理I/Fの来歴確認呼び出し用URL末尾に"/{交換実績記録用リソースID }"と追記することで設定する。
7. 来歴確認の応答を確認し、次のいずれかの処理を行う。
   * HTTPステータスコードが「2xx」以外の場合は、独自Exceptionを発生させる。  
     また、HTTPステータスコードが「2xx」だが、レスポンスから正常に値が取得できない場合は、独自Exceptionを発生させる。
   * HTTPステータスコードが「2xx」の場合、検索結果に取得した検索結果とHTTPステータスコード「200」をレスポンスに設定し、応答を返す。

#### 履歴ID検索

1. HTTPリクエストボディとして、認証トークン、履歴検索用文字列を取得する。
2. 履歴検索用文字列をメッセージコードとともにログ出力する。
3. コンフィグファイルから以下を取得する。コンフィグから情報が取得できない場合、独自Exceptionを発生させる。
   * provenance.json
     1. 来歴管理サービスへのアクセスのベースURL（provenance\_management\_api\_url）
4. 認証トークン、履歴検索用文字列を引数として、来歴管理サービスの履歴検索をPOST通信で呼び出す。履歴検索用文字列はリクエストのbody部に設定する。
5. 履歴検索の応答を確認し、次のいずれかの処理を行う。
   * HTTPステータスコードが「2xx」以外の場合は、独自Exceptionを発生させる。  
     また、HTTPステータスコードが「2xx」だが、レスポンスから正常に値が取得できない場合は、独自Exceptionを発生させる。
   * HTTPステータスコードが「2xx」の場合、検索結果に取得した検索結果とHTTPステータスコード「200」をレスポンスに設定し、応答を返す。

### 出力ログ

独自Exceptionに付与するエラーメッセージおよび出力ログの情報は「詳細設計書\_別紙1\_メッセージ一覧.xlsx」を参照すること。