RESILINK Mid-platform API Documentation

Responsible Editor: UPPA

Version: 1.0

Date: September 2024

Executive Summary

This document provides a detailed overview of how RESILINK Mid-Platform operates, built on the ODEP API. RESILINK Mid-Platform acts as a key integration layer, enhancing and optimizing the functionalities offered by ODEP for various business applications.

This document outline the platform's main features, benefits, and potential use cases across different contexts. Additionally, the document highlights the limitations associated with integrating the ODEP API, offering insight into operational constraints and possible areas for improvement.

Introduction

The RESILINK Mid-platform API is an open-source interface designed to streamline resource exchanges by integrating with ODEP (Orange Decentralized Exchange Platform), an API developed by Orange that leverages blockchain technology to facilitate decentralized exchanges. RESILINK enhances the capabilities of ODEP by providing additional features and tools aimed at developing applications and software that manage resource exchanges and contract management.

The API enables users to interact with offer, assets, asset types, and contracts in a decentralized exchange environment, supporting functionalities such as asset creation, management, and regulatory actions. By building on ODEP's contract-based exchange system, the RESILINK Mid-platform offers a versatile foundation for applications focused on peer-to-peer resource transactions, with an emphasis on transparency and security through blockchain technology.

Bearer Token Authentication System

The API uses a **Bearer Token** system for authentication and authorization. Every request must include a valid token in the header to be processed by the server. Below are the steps to retrieve a token and interact with the API.

Sign In to Retrieve Token

To obtain an authentication token, you must use the following endpoint:

- **URL**: `/v1/users/auth/sign_in`
- **Method**: **POST**
- **Request Body**: The body should include the login credentials of a user already registered in the API.

Example request:

```
{
  "email": "your.email@example.com",
  "password": "yourpassword"
}
```

If the credentials are correct, the API will return a JWT (JSON Web Token). This token must be included in the Authorization header for subsequent requests, formatted as follows:

Authorization: Bearer <token>

Using the Admin Account

If no user account exists, you must use the default administrator account to obtain a token, which can then be used to create or manage other accounts.

- **Username**: `admin`
- **Password**: `admin123`

You can call the sign_in endpoint using these credentials to retrieve the admin token. This administrator account has full access rights over all entities and can manage users, prosumers, asset types, etc.

Creating and Managing User Accounts

Once you have obtained the administrator token, you can use it to create new user accounts or manage existing ones. The admin account is crucial for:

- Creating and managing users (e.g., Prosumers, Regulators).
- Creating contracts.
- Overseeing various data.

For the creation of other entities, we recommend that you use an account other than that of the administrator, as the account is an administrator and not a prosumer or regulator.

API Features

The RESILINK Mid-Platform API enhances its functionality by incorporating several key features:

- **User Management:** The API supports user profiles, which can be categorized as either prosumers or regulators. This dual-user status facilitates diverse interactions within the platform, allowing for tailored experiences based on user roles.
- **Asset Types Management:** Asset types must first be defined before assets can be created. This functionality allows users to create custom asset types, which are crucial for organizing and classifying assets. These types ensure a structured approach to managing the different resources available for exchange on the platform.
- **Asset Management:** Once asset types are defined, users can create, manage, update, and delete assets. Assets represent the resources available for trade, and each asset must be associated with a predefined asset type. This structured relationship between asset types and assets ensures clarity in asset classification and management across the platform.
- **Offer Management:** Offers are created based on existing assets. Once an asset is listed, users can create offers for trade and are essential for initiating transactions. An offer must be linked to a specific asset, and it forms the basis of contracts between two users.
- **Contracts:** Contracts formalize the agreements between users who engage in asset exchange through offers. They manage rental, sale, and purchase transactions, ensuring transparency, tracking, and enforcement of the agreed terms.
- **News Management:** Users can manage news links and accounts, ensuring they stay informed about relevant updates and announcements within the platform.
- **Requests:** This feature serves as a linking mechanism for searching and discovering offers, enhancing user engagement and facilitating asset exchanges.
- **Blockchain Security:** All exchanges and contracts are securely managed via blockchain technology, adding an extra layer of transparency and trust to transactions.

Summary of API Functionality

- **User Management:** Support for managing/creating user profiles.
- **Prosumer & Regulator Management:** Support for managing prosumer and regulator profiles, which are sub-profiles for Users.
- **Asset Management:** Create, update, retrieve, and delete assets, with regulation capabilities.
- **Asset Types:** Manage and categorize various asset types.
- **Contract Management:** Create and oversee contracts for resource exchanges, including rental and sale/purchase agreements.
- **Offer Management:** Create and manage offers for asset trade.
- **News Management:** Handle news links to keep users informed.

- **Request Functionality:** Facilitate searches for offers, enhancing user engagement.
- **Blockchain Security:** Ensure secure and transparent resource exchanges through blockchain technology.

Overview and Usage

User

Description

The User entity represents a user in the system. It contains basic information about the user, such as personal details and account-related metadata. When a new user is created, certain fields are automatically generated by the system, while others are provided during creation.

JSON Template Example


```
{
  "_id": "615f1b28e13e4c0b58d1a87f",
  "userName": "axel.cazaux",
  "firstName": "Axel",
  "lastName": "Cazaux",
  "roleOfUser": "prosumer",
  "email": "axel.cazaux@example.com",
  "password": "032165",
  "provider": "22004",
  "account": "accountId12345",
  "createdAt": "2023-03-16T14:29:22.000Z",
  "updatedAt": "2023-03-16T15:00:00.000Z",
  "phoneNumber": "+33123456789"
}
```


Key Fields

- **_id**: A unique string identifier for the user. This is typically generated by the system.
- **userName**: The username chosen by the user. It must be unique across the system.
- **firstName**: The user's first name.
- **lastName**: The user's last name.
- **roleOfUser**: The role assigned to the user (e.g., "admin", "prosumer", etc.). This field defines the user's permissions and access level in the system.
- **email**: The user’s email address. This must be unique and valid, as it may be used for account verification and communication.
- **password**: The password for the user’s account. It is stored securely using encryption.

- **phoneNumber**: The user's phone number, required for additional verification or communication.
- **provider**: Automatically generated upon user creation, this field represents the storage or service provider tied to the user, typically corresponding to the localhost (e.g., **22004**).
- **account**: A string linking the user to an account, which is created automatically upon user registration.
- **createdAt**: Timestamp indicating the date and time the user was created in the system.
- **updatedAt**: Timestamp representing the last time the user's information was updated.

Prosumer

Description

The Prosumer entity represents a user who can both produce and consume resources within the system. A prosumer can be created when a user is registered. The prosumer's ID matches the user’s username and holds additional information such as balance, sharing account status, job, location, and bookmarked news.

JSON Template Example

```
{
	"id": "axel.cazaux",
	"SharingAccount": 0,
	"balance": 0,
	"job": "Engineer",
	"location": "Paris",
	"bookMarked": [
		"news1_id",
		"news2_id"
	]
}
```

Key Fields

- **Id**: A string which must be the same as the username of the user (e.g., `"axel.cazaux"`).
- **SharingAccount**: A String that indicates the status of the prosumer's sharing account. Default is `"0"` (inactive), and `"1"` means active.
- **Balance**: An Integer that represents the prosumer’s balance. Default is 0. Can be modified after the prosumer is created.

- **Job**: (String) Represents the prosumer's profession or occupation (e.g., `"Engineer"`).
- **Location**: (String) Specifies the prosumer's location (e.g., `"Paris"`).
- **BookMarked**: (List of Strings) Contains the IDs of news articles (from the **News** entity) that the prosumer has bookmarked.

Important Notes

- All prosumer-related actions must be executed by users with an **administrator role**.
- The **balance** field is an **integer** and can be modified by providing a positive or negative value.
- The **bookMarked** field is a list of strings representing **News entity IDs** that the prosumer has bookmarked.
- The **SharingAccount**, **job**, and **location** fields can be modified through their respective endpoints.

Regulator

Description

The Regulator entity represents a special type of user whose role is to regulate specific assetTypes within the system. A regulator can oversee, and ensure compliance with rules for assetTypes assigned to them.

JSON Template Example

```
{
  "id": "john.doe",
  "account": "0xg8ess3g1q8z12gs4ges8846sef7a2",
  "assetTypes": [
        "APPLE",
        "CAR"
  ]
}
```

Key Fields

- **id**: (String) The unique identifier of the regulator, typically the same as the username of the **User** (e.g., `"john.doe"`).
- **account**: (String) The account associated with the regulator. This field is created automatically after the regulator is registered in the system (e.g., `"0xg8ess3g1q8z12gs4ges8846sef7a2"`).
- **assetTypes**: (List of Strings) A list of asset types (by their name) that the regulator is responsible for (e.g., `["APPLE", "CAR"]`).

AssetType

Description

The AssetType entity defines a category of assets on the RESILINK platform. Before creating specific assets, an AssetType must be established to organize and classify the resources being exchanged. The level of detail can vary, and these details are implemented through the specificAttributesModel tag.

JSON Template Example


```
{
  "name": "APPLE",
  "description": "RESILINK",
  "nature": "immaterial",
  "unit": "kg",
  "regulated": false,
  "regulator": "string",
  "sharingIncentive": true,
  "specificAttributesModel": [
          {
          "name": "Color",
          "type": "string",
          "mandatory": "false",
          "hasValueList": "true",
          "valueList": "red,green,yellow"
          },
          {
          "name": "Localisation",
          "type": "geographicalPoint",
          "mandatory": "true",
          "hasValueList": "false",
          "valueList": ""
          }
  ]
}
```


Key Fields

- **Name**: A unique name for the asset type. In this example, it's "APPLE".
- **Description**: A brief description of the asset type. This can be any text that helps identify the asset (e.g., "RESILINK").
- **Nature:** Defines the type and measurability of the asset type. There are three possible values:
    - material: Enumerated items (e.g., 2 laptops, but not 30kg of laptops).

    - immaterial: Measurable items with a fixed unit (e.g., kg but not liters).
    - immaterialNotQuantified: Measurable items where the unit can change or cannot be fixed (e.g., different forms of energy).
- **Regulated:** A boolean (true or false) that indicates whether a regulator is required for this asset type. If true, the regulator field must be populated.
- **Regulator**: A string specifying the regulator if the asset is regulated (only required if **regulated** is set to true).
- **SharingIncentive**: A boolean that indicates whether sharing incentives are applicable to this asset type.
- **SpecificAttributesModel:** Defines additional parameters for the asset type. This field contains a list of attributes with specific characteristics that may be required or optional.
    - **Name:** The attribute name.
    - **Type:** The type of data this attribute will accept. Possible values include:
        - string: A string value (can be restricted by a valueList).
        - numeric: A numeric value (e.g., size or quantity).
        - boolean: A boolean value (e.g., whether the item is organic).
        - listAsset: A list of values.
        - geographicalPoint: Coordinates for geolocation.
    - **Mandatory:** Indicates whether this field is required.
    - **hasValueList**: A boolean that indicates if the attribute has a predefined list of values (e.g., color options for an apple).
    - **valueList**: A list of acceptable values for the attribute, provided as a comma-separated string (e.g., "red,green,yellow"). This field is only relevant if **hasValueList** is set to true.

Example

In the given JSON template, a generic asset type for "APPLE" is defined. The asset type is categorized as immaterial with kg as the unit. The specificAttributesModel requires the prosumer to provide the apple's color, chosen from a predefined list of values (red, green, yellow). Additionally, the prosumer must provide the geographical location of the asset, which is mandatory.

Asset

Description

An Asset represents a specific resource that can be traded or exchanged on the RESILINK platform. Each asset must be linked to a predefined AssetType and owned by a registered user (prosumer). Assets are created with specific attributes and can be involved in transactions such as sale/purchase or rental agreements.

JSON Template Example

```
{
  "name": "My apple batch",
  "description": "I have a nice batch of freshly harvested apples",
  "assetType": "APPLE",
  "owner": "axel.cazaux",
  "transactionType": "sale/purchase",
  "totalQuantity": "30",
  “images”: “”,
  “unit”: “g”,
  "specificAttributes": [
        {
              "attributeName": "Color",
              "value": "yellow"
        },
        {
              "attributeName": "Localisation",
              "value": "<43.31336566835476,-0.365068669622542>"
        }
  ]
}
```


Key Fields

- **Name:** A freely chosen name for the asset.
- **Description:** A brief description of the asset, which can also be customized.
- **AssetType:** The type of asset, which must correspond to an existing **AssetType** defined earlier (e.g., “APPLE”).
- **Owner:** The username of the prosumer who owns the asset.
- **TransactionType:** Specifies the type of transaction in which the asset will be involved. Only two values are allowed:
    - sale/purchase
    - rent
- **TotalQuantity:** Required only for assets of **material** or **immaterial** types. This represents the total available quantity of the asset (e.g., 30 kg of apples).
- **SpecificAttributes:** A set of key-value pairs that define the specific attributes of the asset, as defined by the associated **AssetType**.
    - **attributeName:** Corresponds to the name of the specific attribute as defined in the **AssetType**.
    - **Value:** The actual value for this attribute (e.g., “yellow” for the color, geographical coordinates for location).

An **id** will be automatically generated for each asset upon creation.

Offer

Description

An Offer represents a proposal made by a prosumer to trade. Offers are tied directly to assets and contain specific details about availability, pricing, and conditions of the trade. The offer can involve the sale, purchase, or rental of the asset.

JSON Template Example

```
{
  "offerer": "axel.cazaux",
  "assetId": 24,
  "beginTimeSlot": "2023-03-20T15:46:02.674Z",
  "endTimeSlot": "2023-03-25T15:46:02.675Z",
  "validityLimit": "2023-03-25T15:46:02.675Z",
  "offeredQuantity": 30,
  "price": 2,
  "deposit": 10,
  "cancellationFee": 5,
  "rentInformation": {
        "delayMargin": 10,
        "lateRestitutionPenality": 40,
        "deteriorationPenality": 50,
        "nonRestitutionPenality": 100
  }
}
```

Key Fields

- **Offerer:** The username of the prosumer creating the offer.
- **AssetId:** The unique identifier (ID) of the asset being offered, which is automatically generated when the asset is created.
- **BeginTimeSlot:** The start date and time from which other users (prosumers) can begin making requests to this offer. Format is a string in date-time format (e.g., 2023-03-20T15:46:02.674Z).
- **EndTimeSlot:** The end date and time at which prosumers can no longer make requests for this offer. This field is optional for assets that are not of the **immaterial** type or not for **material rent**.
- **ValidityLimit:** The expiration time for the offer, beyond which it will no longer be visible, even if there is no available quantity.
- **OfferedQuantity:** The quantity of the asset the prosumer is offering to sell or trade. This field is required only for **immaterial** assets and must be an integer (e.g., 30).
- **Price:** The price per unit of the asset being offered. It is expressed as an integer (e.g., 2).
- **Deposit:** A percentage amount (expressed as an integer) representing the deposit the buyer must pay.

- **CancellationFee:** The penalty fee (integer) that the buyer must pay if they cancel the request after it's accepted.
- **RentInformation:** This field is required only for assets of the **material rent** type. It contains additional conditions specific to rental transactions, such as:
    - **DelayMargin:** Allowed margin for delayed returns (e.g., 10 minutes).
    - **LateRestitutionPenalty:** Penalty for late returns (e.g., 40 units of currency).
    - **DeteriorationPenalty:** Penalty for damaged assets (e.g., 50 units).
    - **NonRestitutionPenalty:** Penalty for failing to return the rented asset (e.g., 100 units).

The **offerId** is automatically generated and provided by the API when the offer is created.

Request

Description

A Request is created by a prosumer to search for available offers, either based on a specific AssetType or specific Asset(s). It allows users to filter offers based on various criteria, including time slots, transaction types, price, quantity, and asset attributes.

JSON Template for Request Creation

There are two main methods to create a Request: searching by AssetType or searching by specific Offer IDs.

Search by AssetType Example:

```
{
 "requestor": "axel.cazaux",
 "beginTimeSlot": "2023-03-20T15:46:02.611Z",
 "endTimeSlot": "2023-03-22T10:28:50.612Z",
 "validityLimit": "2023-03-23T10:28:50.612Z",
 "transactionType": "sale/purchase",
 "assetTypes": [
        {
         "assetTypeName": "APPLE",
         "maximumPrice": 50,
         "maximumDeposit": 30,
         "requestedQuantity": 10,
         "requestedSpecificAttributes": [
         {
          "attributeName": "Color",
          "value": "yellow",
          "comparisonType": "contains"
         }
         ]
        }
```

```
  ]
}
```

Search by Offer IDs Example:

```
{
  "requestor": "axel.cazaux",
  "beginTimeSlot": "2023-03-20T15:46:02.611Z",
  "endTimeSlot": "2023-03-22T10:28:50.612Z",
  "validityLimit": "2023-03-23T10:28:50.612Z",
  "transactionType": "sale/purchase",
  "offerIds": [
        10
  ]
}
```

Key Fields

- **Requestor:** The username of the prosumer creating the request.
- **BeginTimeSlot:** The starting time for when the request is valid. This field follows a string date-time format (e.g., `2023-03-20T15:46:02.611Z`).
- **EndTimeSlot:** The end time for the request, after which it will no longer be considered valid for new offers.
- **ValidityLimit:** The final deadline for the request, indicating the point after which it will expire.
- **TransactionType:** Defines the nature of the transaction, which can take two values:
    - **sale/purchase**
    - **rent**
- **AssetTypes:** (Only required when searching by asset types) A list of asset types being searched. Each entry includes:
    - **AssetTypeName:** The name of the asset type (e.g., `"APPLE"`).
    - **MaximumPrice:** The maximum price the prosumer is willing to pay for the asset.
    - **MaximumDeposit:** The maximum deposit the prosumer is willing to commit to.
    - **RequestedQuantity:** The minimum quantity the prosumer is looking to acquire. The available quantity in the offer must exceed or equal this value.
    - **RequestedSpecificAttributes:** (Optional) This field allows for specific filtering of assets based on their attributes, such as color, location, or other predefined characteristics.
- **OfferIds:** (Only required when searching by specific offers) A list of offer IDs to specifically target certain offers.

RequestedSpecificAttributes Fields

- **AttributeName:** The name of the specific attribute as defined in the **AssetType**.
- **Value:** The value being searched for. This field's format depends on the **ComparisonType**.

- **ComparisonType:** Defines how the **Value** field will be compared to the values in the offers. Possible values are:
    - `contains`: Checks if the offer's value contains the specified string.
    - `==`: Checks if the value matches exactly.
    - `>=`: Checks if the value is greater than or equal.
    - `<=`: Checks if the value is less than or equal.
    - `>`: Checks if the value is greater than.
    - `<`: Checks if the value is less than.
    - `between`: Checks if the value lies between two numeric values.
    - `in`: Checks if the value is included in a list.
    - `conj`: Checks for a conjunction of values.
    - `disj`: Checks for a disjunction of values.
    - `in(circle)`: Checks if the geographical point is within a defined circle (requires coordinates and radius).
    - `in(rectangle)`: Checks if the geographical point is within a defined rectangle (requires coordinates for the vertices).

Response from API

When a **Request** is successfully created or queried, the API will return a standardized response in the following format:

```
{
  "requestId": integer,
  "message": "string",
  "availableOffersCount": integer,
  "availableOffersIds": [
        integer
  ]
}
```

- **requestId:** The unique ID assigned to the created request.
- **message:** Any status message or additional information.
- **availableOffersCount:** The number of available offers that match the request.
- **availableOffersIds:** A list of IDs of the offers that match the search criteria.

Contract

Description

A **Contract** represents the agreement between two prosumers—one offering an asset and the other requesting it—allowing for the transaction and management of deposits, deliveries, and payments. Contracts are created when a valid **Offer** and **Request** are matched, and they track the state and completion of the transaction.

Creating a Contract

To create a **Contract**, you need to provide the **offerId** and **requestId** in the following JSON template:

Contract Creation JSON:

```
{
  "offerId": 0,
  "requestId": 0
}
```

If both the **Offer** and **Request** exist and match the transaction conditions, a contract will be created.

Stored Contract Format

Once the contract is created, it is stored in the following format:
Stored Contract JSON Example:

```
{
  "state": "contractCreated",
  "creationDate": "2023-03-16T14:29:22.000Z",
  "transactionType": "sale/purchase",
  "offerer": "axel.cazaux",
  "offer": "67",
  "asset": "33",
  "requester": "axel.cazaux",
  "request": "240",
  "price": 1,
  "deposit": 3,
  "cancellationFee": 10,
  "quantityToDeliver": 30,
  "deliveredQuantity": 0,
  "consumedQuantity": 0,
  "beginTimeSlot": "2023-03-16T15:23:11.000Z",
  "endTimeSlot": "2023-03-16T18:22:11.000Z",
  "effectiveBeginTimeSlot": "1970-01-01T00:00:00.000Z",
  "effectiveEndTimeSlot": "1970-01-01T00:00:00.000Z",
  "rentInformation": {
        "delayMargin": 0,
        "lateRestitutionPenality": 0,
        "deteriorationPenality": 0,
        "nonRestitutionPenality": 0
  }
}
```

Key Fields

- **State:** The current state of the contract. This is the most important field for tracking the progress of the contract.
- **CreationDate:** The date and time the contract was created.
- **TransactionType:** Either **sale/purchase** or **rent**.
- **Offerer:** The username of the prosumer who offered the asset.
- **Offer/Request:** The IDs of the associated offer and request.
- **Price, Deposit, CancellationFee:** Financial terms agreed upon in the contract.
- **QuantityToDeliver, DeliveredQuantity, ConsumedQuantity:** Fields that track the delivery and consumption of the asset (for measurable assets).
- **BeginTimeSlot, EndTimeSlot:** The scheduled start and end of the contract's transaction period.
- **EffectiveBeginTimeSlot, EffectiveEndTimeSlot:** The actual start and end times of the transaction, which may differ from the scheduled times.
- **RentInformation:** Penalties and margin details, relevant only for rental contracts.

Contract State Updates

The **state** field is crucial for managing and updating the status of the contract. The values it can take depend on the nature of the asset:

For Immaterial Assets state can take 3 values for updates:

1. **beginDelivery**
2. **endDelivery**
3. **endOfConsumption**

To update the contract for immaterial assets, you must provide the following JSON:

```
{
  "state": "endDelivery",
  "quantity": 30
}
```

- **Quantity**: Required only for measurable immaterial assets (e.g., if the asset is measured in units).

For Material Assets, state can take 3 values for updates:

1. **assetDeliveredByTheOfferer**
2. **assetReceivedByTheRequestor**
3. **assetNotReceivedByTheRequestor**

To update the contract for material assets, use this JSON:

```
{
  "state": "assetReceivedByTheRequestor"
}
```

For Rental Contracts, state can take 6 values for updates:

1. **assetDeliveredByTheOfferer**
2. **assetReceivedByTheRequestor**
3. **assetNotReceivedByTheRequestor**
4. **assetReturnedByTheRequestor**
5. **assetReturnedToTheOfferer**
6. **assetNotReturnedToTheOfferer**

To update a rental contract, use this JSON template:

```
{
  "state": "assetReturnedToTheOfferer",
  "deterioration": "true",
  "delayPeriod": 1
}
```

- **Deterioration:** Indicates whether the asset has been returned in worse condition.
- **DelayPeriod:** The number of delayed days (required if deterioration or delay applies).

Contract Cancellation

Contracts can be canceled, returning the related **Offer** and **Request** to their available state.

- **Before Contract Execution:** If the cancellation occurs before the contract execution, the state will be updated to **cancelled**, and the initiator will be charged the **cancellationFee**, which will be credited to the other party.
- **After Contract Execution:** If canceled after execution:
    - If canceled by the **offerer**, the state will be updated to **endDelivery**.
    - If canceled by the **requestor**, the state will be updated to **endOfUse**.

For **immaterial assets**, the cancellation allows adjusting payment based on the **effective quantity** delivered or consumed. For **non-measurable immaterial assets**, the payment will be adjusted based on the effective period.

Example of Contract Cancellation (Before Execution)

```
{
  "state": "cancelled"
}
```

Example of Contract Cancellation (After Execution, by Offerer)

```
{
  "state": "endDelivery",
  "quantity": 20
}
```

This indicates the end of delivery for an immaterial asset, and the payment is adjusted according to the effective quantity (e.g., 20 units were delivered).