ConfigCore Service

Overview

This config core service maintains the ***master metadata and configuration*** for all the **Seller Central Platform** services.

Key Features

1. **Master Metadata Management:**
   - The service efficiently manages and organizes master metadata, ensuring accurate and up-to-date information across the system.
2. **Configuration Management:**
   - ConfigCore excels in handling configurations for various components, allowing for flexible and dynamic adjustments as needed.
3. **Seller Central Platform Integration:**
   - Specifically designed for seamless integration with Seller Central Platform services, ensuring harmonious operation and data consistency.
4. **Centralized Control:**
   - The service provides a centralized control point for overseeing and modifying critical configurations and master data elements.
5. **Versioning and Auditing:**
   - ConfigCore incorporates versioning and auditing capabilities, allowing for tracking changes and maintaining a comprehensive history of configurations and master data.
6. **Security Measures:**
   - Implements robust security measures to protect sensitive master data and configuration information from unauthorized access or manipulation.
7. **Cross-Service Consistency:**

- Ensures consistency in master data and configurations across various Seller Central Platform services, promoting a unified and coherent environment.

Schema

Config-Schema

```
{
    "config-name": "",                    // Required: Name of the
configuration.
    "notification-topic": "string",
    "config-type": "JSON/YAML",            // Required: Type of
configuration, either "JSON" or "YAML".
    "config-schema": {                    // Required: JSON/XSD schema for
the configuration.
        [Refer: JSON Config-Schema Contract]
    },
    "config-data": {                      // Required: Actual configuration
data.
        [Refer: Config-Data]
    }
}
```

Explanation:

- `"config-name"`: Represents the name of the configuration. It is a required field.
- `"notification-topic"`: Represents the string indicating the notification topic.
- `"config-type"`: Indicates the type of configuration, which can be either "JSON" or "YAML". It is a required field.
- `"config-schema"`: Represents the JSON/XSD schema for the configuration. It is a required field.
- `"config"`: Represents the actual configuration data. It is a required field.

JSON Config-Schema Contract: *(e.g.: EWayBillConfiguration Contract)*

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "STATE_CODE": { "type": "integer" },
    "COUNTRY": { "type": "string" },
    "EWB_INPUT_REQUIRED": { "type": "string" },
    "EWB_OUTPUT_REQUIRED": { "type": "string" },
    "EWB_REGION": { "type": "string" },
    "EWB_STATE_DESCRIPTION": { "type": "string" },
    "FROM_DATE": { "type": "string", "format": "date" },
    "NET_VALUE": { "type": "number", "format": "float" },
    "REGION": { "type": "string" }
  },
  "required": [
```

```
    "STATE_CODE",
    "COUNTRY",
    "EWB_INPUT_REQUIRED",
    "EWB_OUTPUT_REQUIRED",
    "EWB_REGION",
    "EWB_STATE_DESCRIPTION",
    "FROM_DATE",
    "NET_VALUE",
    "REGION"
  ]
}
```

Config-Data

```
{
  "STATE_CODE": 35,
  "COUNTRY": "IN",
  "EWB_INPUT_REQUIRED": "X",
  "EWB_OUTPUT_REQUIRED": "X",
  "EWB_REGION": "AN",
  "EWB_STATE_DESCRIPTION": "ANDAMAN AND NICOBAR",
  "FROM_DATE": "25-05-18",
  "NET_VALUE": 50000.00,
  "REGION": "AN"
}
```

Master Data Schema

```
{
  "master-data-name": "",         // Name of the master data.
  "schema": {},                   // JSON schema for each row, e.g.,
{"stateName": "string", "state-code": number}.
  "collection-name": "",          // Name of the collection.
  "redis-map-name": "",           // Redis map name, e.g., state-code-map.
  "redis-key": ""                 // Expression - schema-driven, e.g.,
"stateName".
}
```

Master Data Metadata JSON Structure

This JSON structure defines metadata for managing master data within a system. It includes the following key elements:

- `master-data-name`: The name assigned to the master data, providing a unique identifier for referencing this specific set of data.
- `data-schema`: Describes the structure of each row in the master data. The schema is specified as a JSON object, indicating the expected properties and their respective data types. For example, in the

given comment, it suggests a schema with properties like "stateName" of type "string" and "state-code" of type "number".

- `collection-name`: Specifies the name associated with the collection of master data. This can be used to categorize or group similar types of data.
- `redis-map-name`: Refers to the name of the Redis map associated with this master data. For instance, the comment suggests a Redis map named "state-code-map".
- `redis-key`: Represents an expression, likely used for generating Redis keys. The comment provides an example expression, "stateName", indicating that the Redis key might be driven by the "stateName" property in the master data schema.

JSON Master Schema

```
{
  "master-data-name": "ExampleMasterData",
  "schema": {
    "stateName": "string",
    "state-code": "number",
    "population": "number",
    "capital": "string"
  },
  "collection-name": "StateDataCollection",
  "redis-map-name": "state-code-map",
  "redis-key": "stateName"
}
```

API Design

Config-Schema API Contract

POST - `/config`

Creates a new configuration.

> This is an Admin - Write operation.

Request

```
{
    "config-name": "ExampleConfig",         // Required: Unique name of
the configuration.
    "notification-topic": "exampleTopic",  // Optional: Notification
topic.
    "config-type": "JSON",                  // Required: Type of
configuration, either "JSON" or "YAML".
    "config-schema": {
        // [JSON/YAML schema contract]
```

```
    },
    "config": {
        "property1": "value1",
        "property2": 42
    }
}
```

Response

```
{
    "status": "success",
    "message": "Configuration 'ExampleConfig' created successfully.",
    "config-id": "123456",
    "timestamp": "2024-01-02T12:34:56Z"
}
```

PUT - `/config/{config-name}`

Update an existing configuration.

> This is an Admin - Write operation.

Request

```
{
    "config-name": "ExampleConfig",         // Required: Unique name of
the configuration.
    "notification-topic": "exampleTopic",   // Optional: New notification
topic.
    "config-type": "JSON",                   // Optional: New type of
configuration.
    "config-schema": {
        // Optional: New JSON/YAML schema for the configuration.
    },
    "config": {
        // Verify config with schema (new if present or else compare with
old)
        "property1": "new_value1",
        "property2": 99
    }
}
```

The API responds with a JSON object indicating the status and result of the update operation.

Success Response (HTTP Status Code: 200 OK)

```
{
    "status": "success",
    "message": "Configuration 'ExampleConfig' updated successfully.",
    "config-id": "123456",
    "timestamp": "2024-01-02T12:34:56Z"
}
```

GET - `/config`

Retrieves the list of configuration details for all services.

> Read operation is not restricted to Admin Role.

Success Response (HTTP Status Code: 200 OK)

```
[
    {
        "config-name": "ExampleConfig_1",
        "notification-topic": "exampleTopic_1",
        "config-type": "JSON",
        "config-schema": {
            // [JSON/YAML schema contract]
        },
        "config": {
            "property1": "value1",
            "property2": 42
        }
    },
    {
        "config-name": "ExampleConfig_2",
        "notification-topic": "exampleTopic_2",
        "config-type": "JSON",
        "config-schema": {
            // [JSON/YAML schema contract]
        },
        "config": {
            "property1": "value1",
            "property2": 42
        }
    }
    // Add more configurations as needed
]
```

GET `/config/{config-name}`

Returns configuration basis configuration name

> Read operation is not restricted to Admin Role.

Success Response (HTTP Status Code: 200 OK)

```
{
    "config-name": "ExampleConfig",         // Required: Unique name of
the configuration.
    "notification-topic": "exampleTopic",   // Optional: New notification
topic.
    "config-type": "JSON",                   // Optional: New type of
configuration.
    "config-schema": {
        // Optional: New JSON/YAML schema for the configuration.
    },
    "config": {
        // Verify config with schema (new if present or else compare with
old)
        "property1": "new_value1",
        "property2": 99
    }
}
```

GET `/config/{search-criteria}`

Returns configuration basis configuration name

> Read operation is not restricted to Admin Role.

Success Response (HTTP Status Code: 200 OK)

```
{
    "config-name": "ExampleConfig",         // Required: Unique name of
the configuration.
    "notification-topic": "exampleTopic",   // Optional: New notification
topic.
    "config-type": "JSON",                   // Optional: New type of
configuration.
    "config-schema": {
        // Optional: New JSON/YAML schema for the configuration.
    },
    "config": {
        // Verify config with schema (new if present or else compare with
old)
        "property1": "new_value1",
        "property2": 99
    }
}
```

Master Metadata Schema API Contract

POST `/master-data/meta-data` Creates a new master meta-data.

Request:

```
{
    "master-data-name": "exampleMasterData",
    "schema": {
        "stateName": "string",
        "state-code": "number"
    },
    "collection-name": "exampleCollection",
    "redis-map-name": "state-code-map",
    "ID": "stateName"
}
```

Response

```
{
    "status": "success",
    "message": "Configuration 'exampleMasterData' created successfully.",
    "master-id": "123456",
    "timestamp": "2024-01-02T12:34:56Z"
}
```

PUT `/master-data/meta-data/{master-data-name}`

Request:

```
{
    "schema": [
        {"stateName": "string", "state-code": "number"}
        // Add more schema rows as needed
    ],
    "redis-map-name": "state-code-map",
    "ID": "stateName"
}
```

Response

```
{
    "status": "success",
    "message": "Configuration 'exampleMasterData' updated successfully.",
```

```
    "timestamp": "2024-01-02T12:45:00Z"
}
```

DELETE `/master-data/meta-data/{master-data-name}`

Response

```
{
    "status": "success",
    "message": "Configuration 'exampleMasterData' deleted successfully.",
    "timestamp": "2024-01-02T13:00:00Z"
}
```

GET `/master-data/{meta-data-name}` Retrieves all paginated entries in meta-data.collection-name

Response

```
{
    "status": "success",
    "message": "Entries retrieved successfully.",
    "data": [
        {
            "stateName": "California",
            "state-code": 123
        },
        // Add more entries as needed
    ],
    "pagination": {
        "page": 1,
        "pageSize": 10,
        "totalEntries": 100
    }
}
```

GET `/master-data/{meta-data-name}/{ID}` Retrieve an entry from the master data collection filtered by the provided ID.

Response

```
{
    "status": "success",
    "message": "Entry retrieved successfully.",
    "data": {
        "stateName": "California",
        "state-code": 123
    }
```

```
}
```

API Error Codes

APIs often return specific error codes to indicate the status of a request. Here are some common error codes and their meanings:

400 Bad Request

- **Description**: The request was malformed or invalid.
- **Possible Causes**: Missing or incorrect parameters in the request.

401 Unauthorized

- **Description**: Authentication is required, and the provided credentials are not valid.
- **Possible Causes**: Missing or incorrect authentication token.

403 Forbidden

- **Description**: The authenticated user does not have permission to access the requested resource.
- **Possible Causes**: Insufficient privileges for the authenticated user.

404 Not Found

- **Description**: The requested resource could not be found on the server.
- **Possible Causes**: Incorrect endpoint or resource name.

500 Internal Server Error

- **Description**: An unexpected error occurred on the server.
- **Possible Causes**: Server-side issues or bugs in the API.

Custom Error Codes

- `1001` **Configuration Not Found**
  - **Description**: The specified configuration does not exist.
  - **Possible Causes**: The provided configuration name is incorrect.
- `1002` **Invalid Configuration Data**
  - **Description**: The configuration data does not match the required schema.
  - **Possible Causes**: Incorrect format or missing required properties.