Writing Samples – API Documentation

Writing Sample 1: Oracle Rest Data Services API

Oracle REST Data Services (ORDS) empowers users to manage and monitor Oracle Database operations through an intuitive REST API. Depending on the database's version and configuration, ORDS offers various services, including managing pluggable databases, data export capabilities, and database performance reviews.

Reference Link: ***https://docs.oracle.com/en/database/oracle/oracle-rest-data-services/21.4/orrst/index.html***

REST Endpoints are Get a Database link, Get a Partition

1. Endpoint: Get a Database link

The Get Database Link API provides a RESTful interface to access details about database links used to connect to remote databases. Depending on the role access at runtime, it uses a DBA_DB_LINKS or ALL_DB_LINKS view. A client requires an SQL Administrator or SQL Developer role to invoke this service. This API is essential for database administrators and developers who must programmatically manage and monitor database links.

Request Syntax

To call the Get Partition API, use the following HTTP GET request format:
GET /ords/{workspace}/{schema}/db_links/{db_link_name}

Parameters

- **workspace** (required):
  - Type: string
  - Description: The name of the workspace where the schema is located.
- **schema** (required):
  - Type: string
  - Description: The name of the database schema containing the link.
- **db_link_name** (required):
  - Type: string
  - Description: The name of the database link to retrieve.

Example Request

Structuring a request to retrieve a specific database link:
GET /ords/my_workspace/my_schema/db_links/my_db_link

Response Format

Upon successful execution of the API call, the response will include details about the requested database link in JSON format. The response structure is as follows:

{ "db_link": {

"name": "my_db_link",

"username": "remote_user",

"host": "remote_host", "service_name": "remote_service",

"created": "2025-01-01T00:00:00Z",

"last_updated": "2025-01-20T00:00:00Z"

}

}

Error Handling

If the request cannot be fulfilled due to issues such as a non-existent database link, an error message will be returned in the following format:

{

"error":

{ "code": "404",

"message": "Database link not found."

}

}

Use Cases

- **Database Management**: Retrieve information about existing database links for monitoring and administration.
- **Troubleshooting**: Identify issues with remote connections by examining database link configurations.
- **Integration**: Facilitate integration tasks that require knowledge of remote database connections.

Summary

This API is essential for applications interacting with multiple Oracle databases, providing a streamlined approach to managing and accessing remote data sources.

2. Endpoint: Get a partition

The Get a Partition API in Oracle REST Data Services (ORDS) enables users to access detailed information about a specific partition within a database table. This API retrieves partition details from the DBA_TAB_PARTITIONS or ALL_TAB_PARTITIONS views, depending on the user's role at runtime. Users must have the SQL Administrator or SQL Developer role to utilize this service.

Request Syntax

To call the Get Partition API, use the following HTTP GET request format:

GET /ords/{workspace}/{schema}/{table}/{partition}

Parameters

- **workspace** (required):
  - Type: string
  - Description: The name of the workspace where the schema is located.
- **schema** (required):
  - Type: string
  - Description: The name of the database schema that contains the table.
- **table** (required):
  - Type: string
  - Description: The name of the table from which to retrieve the partition.
- **partition** (required):
  - Type: string
  - Description: The name or identifier of the partition to retrieve.

Example Request

Structuring a request to retrieve a specific partition,

GET /ords/my_workspace/my_schema/my_table/my_partition

Response Format

Upon successful execution of the API call, the response will include details about the requested partition in JSON format. The response structure is as follows:

```
{
"partition": {
 "name": "my_partition", "values": ["value1", "value2"],
"metadata": {
"created": "2025-01-01T00:00:00Z",
"last_updated": "2025-01-20T00:00:00Z"
    }
  }
}
```

Error Handling

If the request cannot be fulfilled due to a non-existent partition, an error message will be returned in the following format:

{

"error": {

"code": "404",

"message":

"Partition not found."

}

}

Use cases

- **Data Retrieval**: Access specific partition metadata for reporting and analysis.
- **Data Management**: Manage partitions within large datasets for optimization and performance.
- **ETL Processes**: Integrate with ETL workflows that require access to specific partitions for processing.

Summary

This API is essential for applications leveraging partitioned tables in Oracle databases, facilitating efficient data management and retrieval operations.

Writing Sample 2: ATONARP Rest Data Services (Patient's Database)

ATONARP REST Data Services enables developers to retrieve patient information from the database. It offers crucial documentation and resources that facilitate effective interaction with the API designed explicitly for managing patient data.

Reference link: *https://www.atonarp.com*

Introduction

These APIs facilitate operations related to patient records, including creating, retrieving, updating, and deleting patient information.

Authentication

- **API Key:** A unique key provided to each developer must be included in the header of every request.
- **OAuth:** A more secure method that allows users to authorize applications without sharing their credentials.

Getting Started

To authenticate, include the following in your request headers:

Authorization: Bearer {your_api_key}

Endpoints

The API consists of several endpoints that allow various operations on patient data. Below is a summary of the key endpoints:

Endpoint	Method	Description
'/patients'	POST	Create a new patient record
'/patients'	GET	Retrieve a list of all patients
'/patients/{id}'	GET	Retrieve details of a specific patient
'/patients/{id}'	PUT	Update an existing patient record
'/patients/{id}'	DELETE	Delete a specific patient record

1. Create a new patient record

Endpoint

POST/api/patients

Description

Creates a new patient record in the database.

Request and Response Format Example

Request

To create a new patient record, send a POST request to /patients with the following JSON body.

```
{
  "name":
  "vinay",
```

```
  "dob": "1986-04-05",
  "gender": "male",
 "contact": {
      "phone": "1234567890",
      "email": "vinayspk6@gmail.com"
 }
}
```


Response

A successful response will return a status code of 201 and the created patient object.


```
{
"id": "1",
"name": "vinay",
"dob": "1990-01-01",
"gender": "male",
"contact": {
     "phone": "1234567890",
     "email": "vinayspk6@gmail.com"
        }
}
```


Possible error Messages

The API employs standard HTTP status codes to indicate the success or failure of requests:

- **400 Bad Request**: Invalid patient data provided. Please check the input fields.
- **409 Conflict**: A patient with this identifier already exists.
- **500 Internal Server Error**: An unexpected error occurred on the server.

2. Retrieve a list of all patients

Endpoint

To retrieve a list of all patients, use the following endpoint:
GET/api/patients?limit={number}

Description

Retrieves a list of patients, limited by the specified number, to prevent performance issues.

- 'page' (optional): The page number for pagination.
- 'limit' (optional): Number of records per page (default is 50).

Request and Response Format Example

Request

GET https://api.yourhealthsystem.com/v1/patients?page=1&limit=50

Response:

Each resource includes:

- **id:** Patient's unique identifier.
- **name:** Patient's name.
- **gender:** Patient's gender.
- **Birth date:** Patient's date of birth.
- **address:** Patient's address.


```
{
  "patients": [
    {
      "id": "1",
      "name": "Praveen",
      "gender": "male",
      "birthdate": "1980-01-01",
      "address": "23 Main RNagar, INDIA"
    },
    {
      "id": "2",
      "name": "Payal",
       "gender": "Female",
       "birthdate": "1990-05-15",
       "address": "33 Cross MG Road, INDIA"
    }
  ],
  "total": 100,
  "page": 1,
  "limit": 50
}
```


Possible Error Messages:

- 404 Not Found: "No patients found in the database."
- 400 Bad Request: "Invalid limit parameter. Please provide a valid number.

3. Retrieve Specific Patient Details

Endpoint

To retrieve detailed information about a specific patient, use the following
GET /patients/{id}

Request and Response Format Example

Request

GET https://api.yourhealthsystem.com/v1/patients/1

Response


```
{
    "id": "1",
    "name": "Praveen",
    "gender": "male",
    "birthdate": "1980-01-01",
    "medicalHistory": [
      {
        "condition": "Hypertension",
        "diagnosedDate": "2015-06-01"
      }
    ],
    "currentMedications": [
      {
        "medicationName":
        "Lisinopril",
        "dosage": "10mg"
      }
    ]
}
```


Possible Error Messages:

- 404 Not Found: Patient record not found in the database.
- 400 Bad Request: Invalid patient demographics
- 500 Server Error: An unexpected error occurred on the server. Please try again later.

3. Delete a Patient

Endpoint

To delete a specific patient from the database, use the following endpoint:
DELETE /patients/{id}

Description

Deletes a specific patient record from the database.

Parameters:

{id} (required): The unique identifier of the patient you wish to delete.

Request and Response Format Example

Request

DELETE https://api.yourhealthsystem.com/v1/patients/1

Response (Success)


```
{
   "result": "deleted",
   "id": "1"
}
```


Response (Conflict)


```
{
"resourceType":  "OperationOutcome",
"issue": [
    {
       "severity": "error",
       "code": "processing",
       "diagnostics": "Unable to delete Patient/1 because at least one resource references this
         resource."
      }
  ]

}
```


Possible Error Messages:

- 200 OK: Indicates successful deletion of the patient.
- 410 Gone: The patient resource was successfully deleted and is no longer available.
- 404 Not Found: The specified patient ID does not exist.
- 409 Conflict: Unable to delete the patient due to existing references from other resources.