Oracle Hospitality Integration Platform

Revenue Management Systems (RMS) Implementation Guide

Release 25.1
G29726-02
August 2025

ORACLE

Oracle Hospitality Integration Platform Revenue Management Systems (RMS) Implementation Guide, Release 25.1

G29726-02

Contents

Preface

Oracle Hospitality Integration Cloud Service and OPERA Cloud Foundation users are authorized to access the following modules and features:

- Oracle Hospitality Integration Platform including Oracle Hospitality Developer Portal and Hospitality REST APIs.

The Oracle Hospitality Developer Portal enables users to discover Oracle Hospitality APIs, subscribe to Oracle Hospitality APIs, and to get the necessary information to consume them.

The Oracle Hospitality Developer Portal's APIs page provides information about published Oracle Hospitality APIs. Here you can find and evaluate Oracle Hospitality APIs to use with your applications.

After you discover the Oracle Hospitality APIs that you want to use, register an application and then register those APIs to your application.

To view more details about the Oracle Hospitality Developer Portal, on the user menu drop-down, click the drop-down, and select **About Developer Portal**.

Purpose

This guide describes the common use cases and workflows that represent the industry standards and expectations for RMS integrations.

Audience

This guide is intended for customers and partners who develop applications with the Oracle Hospitality Integration Platform.

Documentation

Oracle Hospitality product documentation is available on the Oracle Help Center at https://docs.oracle.com/en/industries/hospitality/.

Revision History

Date	Description of Change
April 2025	Initial publication
August 2025	Formatting changes

1

Business Context

> **Note**
>
> API usage and capabilities are subject to change. For more information on API versioning, see the Versioning topic in the Oracle Hospitality Integration Platform User Guide.
>
> For the most comprehensive and up-to-date information, including additional query parameters and detailed descriptions, always refer to the latest API specifications in GitHub. This will ensure that you are accessing the most current and complete set of features available.

The use of Revenue Management Systems has become increasingly common for properties seeking to maximize the value of each room. A Revenue Management System is a specialized software tool used by hotels, resorts, and other accommodation providers to maximize revenue and profitability. It does this by analyzing a variety of data points to optimize room rates, inventory availability, and overall sales strategies. By leveraging advanced data analytics and real-time market insights, an RMS enables hotels and other accommodation providers to make informed decisions, optimize their operations, and enhance the guest experience. This not only leads to increased revenue, but also ensures long-term competitiveness in the dynamic hospitality market.

Revenue Management Systems typically extract detailed data from OPERA Cloud, including reservations, blocks, rates, and inventory. These systems also update data in OPERA Cloud, such as rates, inventory, and restrictions. The Oracle Hospitality Integration Platform (OHIP) offers three distinct methods for data extraction:

1. Asynchronous Property APIs
2. Synchronous Property APIs
3. Polling or Streaming Business Events Property APIs

This guide describes the common use cases and workflows that represent the industry standards and expectations for RMS integrations.

Use Case	Description
Reservation Management	Retrieve past and future reservation information from a property.
Block Management	Retrieve summary information on Block Allocations for a property.
Rate Management	Retrieve/Update Daily Rate Plan Schedules by posting them to OPERA Cloud.
Inventory Management	Retrieve and update inventory for a given property.
Restrictions Management	Retrieve, create, or update restrictions at various levels for a given property.

2 Prerequisites

Table 2-1 Required Tools

Tool	Description	Links
Github	A repository containing both Oracle Hospitality Property APIs specifications and accompanying Postman Collections.	GitHub Repository
Postman Collection	You can download and use the Postman Collections associated to the use cases explained in this guide.	Postman Collection

Table 2-2 Environments and Modules

OPERA Cloud Platform Module	Description	Minimum Version
Oracle Hospitality Integration Platform (OHIP)	For customers OHIP is included in the subscription to OPERA Cloud Foundation. Partners need a subscription to Oracle Hospitality Integration Cloud Service	The latest released version. OHIP is a single version product.
OPERA Cloud	The customer must have a subscription to OPERA Cloud Foundation	23.5.4.0

Getting Started

To get started, you must perform the following tasks:

- Onboarding to the Oracle Hospitality Integration Cloud Service
- Subscribe to Oracle Hospitality APIs
- Register and Manage Applications

3

Using the OPERA Cloud APIs

When utilizing the OPERA Cloud APIs, you can choose from the following options based on your specific use cases:

- **Synchronous APIs** — The synchronous property APIs are designed to cater to a specific use case in OPERA Cloud by operating on a request-reply basis and demanding immediate responses. For instance, you can use the getReservation API to fetch detailed reservation data. For detailed implementation guidelines on using the synchronous APIs, see Calling Oracle Hospitality Property APIs.
- **Asynchronous APIs** — The asynchronous property APIs are designed for revenue management systems, allowing retrieval and updating of bulk data such as inventory, restrictions, and room rates in OPERA Cloud. For detailed implementation guidelines on using the asynchronous APIs, see Using the Asynchronous APIs.
- **Business Events** — A business event is an event (that is, an update) that happens to a resource, for example, a reservation. For detailed implementation guidelines on using the business events, refer to the Business Events topics.

4
RMS Use Cases

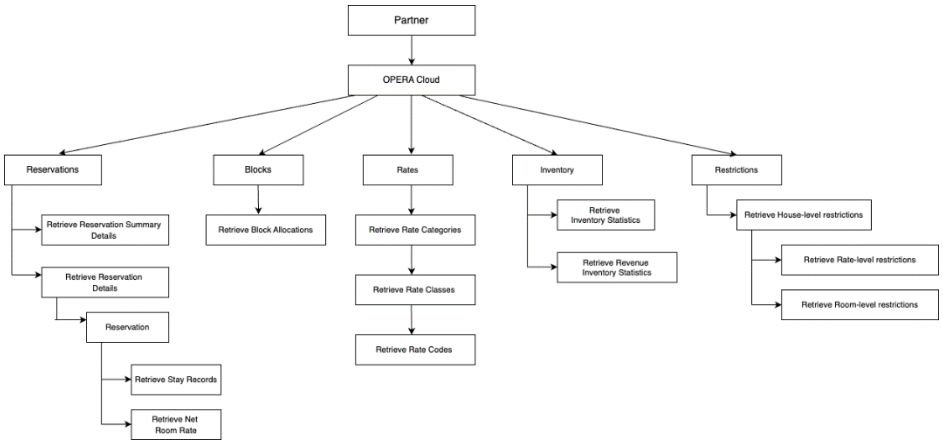
Workflow Overview

There are two main types of RMS Data flows:

1. **Initial data load/sync**:

   Fetching the relevant data for your revenue management system using business events or APIs that can act as a base/snapshot for your analysis. For example, you can fetch bulk reservation and block and inventory data to get started with your analysis.

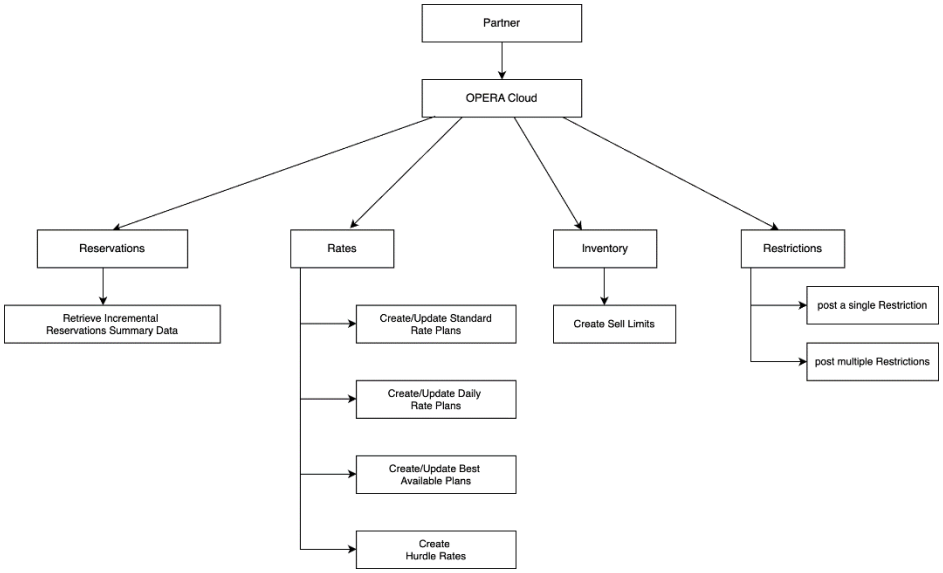
Figure 4-1 Initial Data Load/Sync Workflow



2. **Daily data collection and update**:

   Utilize this flow for your day-to-day workflow to fetch/update incremental data in OPERA Cloud. For example, you can only fetch the updated reservations from the system This also helps to limit the number of requests on your RMS and OPERA Cloud.

Figure 4-2 Daily Data Collection and Update Workflow



Reservation Management

Since rooms can be associated with both reservations and business blocks (which have distinct data models), the methods for retrieving relevant data for each are outlined in different sections in this guide. You can retrieve past and future reservation information for a hotel from OPERA Cloud.

Use Case	Description	Operation
Retrieve summary of reservations activity on a daily basis	You can retrieve a daily summary of reservations for a property by given dates. This can be useful for an overview of reservation activity on a daily basis.	getReservationsDailySummary (asynchronous)
Retrieve detailed reservation information	You can retrieve detailed information about reservations stored in the OPERA Cloud system. This can be useful to analyze reservation data for forecasting and pricing strategies.	getReservations (synchronous)
Retrieve Reservation Rate Break Down	You can retrieve summary rate information based on the given rate plan code, room type, number of adults, and specified date range. This can be useful to analyze pricing trends and set competitive rates.	getRateInfo
Retrieve Reservation Stay Records	Fetches detailed records of a guest's stays at a hotel. This can be useful for analyzing guest history and behavior patterns.	getStayHistory

1. Retrieve summary of reservations activity on a daily basis using getReservationsDailySummary

You must use the asynchronous operation getReservationsDailySummary to capture the daily summary of reservations. This summary captures the key elements of the reservation and stay details, providing an overview of the booking, guest information, stay details, and revenue generated. It can be used for operational planning, revenue management, and reporting purposes.

Business Logic	Comments
Retrieving the initial Reservation Summary Data using start/endDate	The reservation summary data can be requested using startDate/endDate to fetch the reservation summary data for a given date range: • **startDate**: The beginning date to filter reservations. • **endDate**: The ending date to filter reservations. Other critical parameters include: • **hotelId**: Specifies the hotel for which reservations are being queried. • **reservationId**: Retrieves a specific reservation by its unique ID. • **startDate**: The beginning date to filter reservations. • **endDate**: The ending date to filter reservations. • **status**: Filters reservations based on their current status. • **guestName**: Filters reservations by the guest's name. • **roomType**: Filters reservations by the type of room booked. • **rateCode**: Filters reservations based on the rate code applied.
Retrieving the incremental Reservation Summary Data using startLastModifiedDate/endLastModifiedDate	The reservation summary data can be requested using startLastModifiedDate/endLastModifiedDate to fetch the reservation summary data for a given date range: • **startLastModifiedDate**: Filters reservations that have been modified on or after this date. • **endLastModifiedDate**: Filters reservations that have been modified on or before this date.
Key Considerations	• The hotelId parameter is essential to identify which hotel's reservations are being queried. • Keep the date range (startDate and endDate or startLastModifiedDate and endLastModifiedDate) as narrow as possible to reduce the search window to a specific time frame. • Filtering options such as status, guestName, roomType, and rateCode allow for more precise and relevant search results.

Business Logic	Comments
Key Response Data Elements	The key response data elements include: • Hotel Information: – **Hotel ID**: Identifier for the hotel. • Reservation Information: – **Reservation IDs**: Unique identifiers for the reservation, including confirmation and reservation types. – **Status**: Current status of the reservation (for example, Prospect, Confirmed). – **Type**: Type of reservation system used. – **Room Count**: Total number of rooms reserved. • Guest Information: – **Country**: Origin country of the guest. – **Adults**: Number of adults included in the reservation. – **Children**: Breakdown by age groups (1-3, 4-12, 13-17) with counts. • Booking Information: – **Booking and Creation Dates**: Specific dates and times detailing when the booking was made and when the reservation record was created or last modified. • Stay Details: – **Arrival and Departure Dates**: Specific start and end dates of the guest's stay. – **Room Type**: Description of the room type booked. – **Daily Summary**: Includes rate codes, amounts, and market codes. – **Transaction Date**: The specific date of the transaction. • Revenue Information: – **Revenue by Category**: Detailed breakdown of room, food and beverage, and other revenues. – **Total Revenue**: Sum total of all revenue generated from the reservation. – **Tax Details**: Specific tax amounts associated with the booking.

Business Logic	Comments

Sample API request/ response	1. startReservationsDailySummary
	1.1. startReservationsDailySummaryProcess (requested using the parameters startDate/endDate)
	startReservationsDailySummaryProcess (startDate/endDate)
	
	Endpoint:
	POST {{HostName}}/rms/rsv/async/v1/externalSystems/{{ExternalSystemId}}/hotels/{{HotelId}}/reservations/dailySummary
	
	Request Body:
	{"criteria": {
	"hotelId": {{HotelId}},
	"timeSpan": {
	"startDate": "2022-09-02",
	"endDate": "2022-09-02"
	}
	}
	}
	
	Sample Successful Response:
	202 Accepted
	
	1.2. startReservationsDailySummaryProcess (requested using the parameters startLastModifiedDate/endLastModifiedDate)
	startReservationsDailySummaryProcess (startLastModifiedDate/endLastModifiedDate)
	
	Endpoint:
	POST {{HostName}}/rms/rsv/async/v1/externalSystems/{{ExternalSystemId}}/hotels/{{HotelId}}/reservations/dailySummary
	
	Request Body:
	{
	"criteria": {
	"hotelId": {{HotelId}},
	"lastModifiedDate": {
	"startLastModifiedDate": "2023-10-04",
	"endLastModifiedDate": "2023-10-04"
	}
	}
	}
	
	Sample Successful Response:
	202 Accepted

Business Logic	Comments

	2. getReservationsProcessStatus
	getReservationsProcessStatus
	
	Endpoint:
	HEAD {{HostName}}/rms/rsv/async/v1/externalSystems/{{ExternalSystemId}}/hotels/{{HotelId}}/reservations/dailySummary/{{RequestId1}}
	
	Sample Successful Response:
	201 Created
	(You are ready to proceed to Step 3)
	
	3. getReservationsDailySummary
	getReservationsProcessStatus
	
	Endpoint:
	
	GET {{HostName}}/rms/rsv/async/v1/externalSystems/{{ExternalSystemId}}/hotels/{{HotelId}}/reservations/dailySummary/{{RequestId2}}
	
	Sample Successful (200 OK) Response:
	{
	"reservations": [
	{
	"hotelId": "BAA_AUTO",
	"reservationIdList": [
	{
	"id": "1822428",
	"type": "Confirmation"
	},
	{
	"id": 1038537,
	"type": "Reservation"
	},
	{
	"type": "ParentReservation"
	}
	],
	"externalReferences": [],
	"resvStatus": "PROSPECT",
	"sharedYn": "N",
	"arrival": "2022-09-02T00:00:00",
	"departure": "2022-09-03T00:00:00",
	"bookingDate": "2023-04-06T00:00:00",

Business Logic	Comments

	<pre>"resvType": "GDS-SESSION",
	"noOfRooms": "1",
	"guestCountry": "US",
	"createDateTime": "2023-04-06T21:18:19",
	"lastModifiedDate":
	"2023-04-06T21:18:19",
	"createdHotelDateTime":
	"2023-04-06T21:18:19",
	"modifiedHotelDateTime":
	"2023-04-06T21:18:19",
	"children1": 0,
	"children2": 0,
	"children3": 0,
	"origin_of_booking": "FOR",
	"dailySummary": [
	{
	"rateCode": "HAPPY2",
	"rateAmount": "150",
	"rateAmountCurrency": "USD",
	"marketCode": "RETB",
	"roomType": "Z1DLK",
	"bookedRoomType": "Z1DLK",
	"netRateAmount": "150",
	"netRateAmountCurrency": "USD",
	"roomRevenue": "150",
	"roomRevenueCurrency": "USD",
	"fbRevenueCurrency": "USD",
	"otherRevenueCurrency": "USD",
	"totalRevenue": "150",
	"totalRevenueCurrency": "USD",
	"tax": "33.75",
	"taxCurrency": "USD",
	"roomTypeCharged": "Z1DLK",
	"trxDate": "2022-09-02T00:00:00",
	"sourceCode": "FOR",
	"adults": "1"
	}
	]
	}
	]
	}</pre>

2. Retrieve Detailed reservation using getReservations

It is highly recommended that you use the asynchronous operation getReservationsDailySummary for your reservation use cases. However, if you want to fetch detailed information of up to 1000 reservation records including the reservation, guest, room, and rate details, you can use the synchronous operation getReservations.

> **Note**
>
> To save additional costs and calls to the system, only use this synchronous operation when absolutely necessary.

This data is useful to analyze and optimize pricing, availability, and overall revenue strategy. The following is a breakdown of the kind of data it returns and how you can use it.

Business Logic	Comments
Filtering the Reservation Data	You can filter the reservation data based on the following: • **reservationId**: Fetch specific reservation by ID. • **startDate**: Filter reservations starting from this date. • **endDate**: Filter reservations ending on this date. • **status**: Filter by reservation status. • **guestName**: Filter by guest name. • **roomType**: Filter by room type. • **rateCode**: Filter by rate code. • **wildCardSearch**: If you are not sure of the search criteria, you can search by text by entering a sequence of characters. This will search the reservations that match this text in any of the response fields.
Key Response Data Elements	The key response data elements include: • **Reservation ID**: Unique identifier for the reservation. • **Status**: Current status of the reservation (e.g., confirmed, canceled, checked-in). • **Guest Details**: Information about the guest, including name, email, and phone number. • **Room Details**: Details about the room, including room type, room number, check-in and check-out dates. • **Guest Stay Details**: Information about the guest stay details like arrivalDate, departureDate. • **Rate Details**: Information about the rate, including rate code, description, amount, and currency. • **Package Details**: Information about the packages included in the reservation, such as packageCode, packageDescription, and packageAmount. • **Special Requests**: Any special requests made by the guest. • **Payment Details**: Payment method information, including card type and last four digits of the card number.

Business Logic	Comments

Sample API request/ response	getReservations
	getReservations (wild card search)
	
	Endpoint:
	{{HostName}}/rsv/v1/hotels/{{HotelId}}/reservations?
	text=jones
	
	Sample Successful (200 OK) Response:
	"getReservations": {
	"response": {
	"reservations": {
	"reservationInfo": [
	{
	"reservationIdList": [
	{
	"id": "236552",
	"type": "Reservation"
	},
	{
	"id": "379252",
	"type":
	"Confirmation"
	},
	{
	"id": "5735646",
	"type":
	"ExternalReference"
	}
	],
	"roomStay": {
	"arrivalDate":
	"2020-05-08",
	"departureDate":
	"2020-05-09",
	"originalTimeSpan": {
	"startDate":
	"2020-05-08",
	"endDate":
	"2020-05-09"
	},
	"expectedTimes": {
	
	"reservationExpectedArrivalTime": "2020-05-08
	19:17:25.0",
	
	"reservationExpectedDepartureTime": "2020-05-09
	13:30:00.0"
	},
	"adultCount": "2",
	"childCount": "1",
	"roomClass": "ALL",

Business Logic	Comments


```
                                    "roomType": "STDQ",
                                    "numberOfRooms": "1",
                                    "roomNumber": "2010",
                                    "ratePlanCode":
"UPGRADE",
                                    "rateAmount": {
                                        "amount": "100",
                                        "currencyCode": "USD"
                                    },
                                    "rateSuppressed": false,
                                    "fixedRate": false,
                                    "guarantee": {
                                        "guaranteeCode":
"CHECKED IN",
                                        "shortDescription":
"Checked In"
                                    },
                                    "marketCode": "CORP",
                                    "sourceOfBusiness":
"CWEB",
"sourceOfBusinessDescription": "Website",
                                    "balance": {
                                        "amount": "115.55",
                                        "currencyCode": "USD"
                                    },
                                    "roomTypeCharged":
"STDQ",
                                    "roomNumberLocked":
false,
                                    "pseudoRoom": false
                                },
                                "reservationGuest": {
                                    "givenName": "Sarah",
                                    "middleName": "F",
                                    "surname": "Jones",
                                    "phoneNumber": "415 555
0100 ",
                                    "email":
"Sarah@example.com",
                                    "language": "E",
                                    "vip": {
                                        "vipCode": "GOLD",
                                        "vipDescription":
"Gold"
                                    },
                                    "address": {
                                        "cityName": "Redwood
Shores",
                                        "postalCode":
"94065",
```

Business Logic	Comments


```
                                    "state": "CA",
                                    "country": {
                                        "code": "AU"
                                    }
                                },
                                "anonymization": {},

"guestLastStayInformation": {
                                    "lastStayRoom": "304"
                                },
                                "guestRestricted": false,
                                "nameType": "Guest",
                                "id": "288106",
                                "type": "Profile"
                            },
                            "reservationPaymentMethod": {
                                "paymentMethod": "CA"
                            },
                            "displayColor": "GREEN",
                            "reservationIndicators": [
                                {
                                    "indicatorName":
"EXTERNALREFERENCES",
                                    "count": "1"
                                },
                                {
                                    "indicatorName":
"PREFERENCE",
                                    "count": "3"
                                },
                                {
                                    "indicatorName":
"PROFILENOTE",
                                    "count": "2"
                                },
                                {
                                    "indicatorName":
"PACKAGEITEM",
                                    "count": "1"
                                },
                                {
                                    "indicatorName":
"OPENFOLIO"
                                },
                                {
                                    "indicatorName":
"STAYREVENUE"
                                },
                                {
                                    "indicatorName":
"HISTORYFUTURE",
```

Business Logic	Comments


```
                                    "count": "3"
                                },
                                {
                                    "indicatorName":
"COMMUNICATION",
                                    "count": "6"
                                },
                                {
                                    "indicatorName":
"ASSOCIATEDPROFILES",
                                    "count": "1"
                                }
                            ],
                            "sourceOfSale": {
                                "sourceType": "PMS",
                                "sourceCode": "HOTEL1"
                            },
                            "waitlist": {},
                            "advanceCheckIn": {
                                "advanceCheckedIn": true,
                                "expectedReturnTime":
"2020-05-09 15:00:00.0"
                            },
                            "hotelId": "HOTEL1",
                            "hotelName": "Hotel One",
                            "roomStayReservation": true,
                            "createDateTime":
"2020-01-09 01:02:48.0",
                            "lastModifyDateTime":
"2020-02-26 03:00:08.0",
                            "reservationStatus":
"CheckedOut",
                            "computedReservationStatus":
"CheckedOut",
                            "walkInIndicator": true,
                            "commissionPayoutTo": "None",
                            "paymentMethod": "CA",
                            "preRegistered": false,
                            "openFolio": true,
                            "allowMobileCheckout": false,
                            "optedForCommunication":
false
                        }
                    ],
                    "totalPages": "1",
                    "offset": "20",
                    "limit": "20",
                    "hasMore": true,
                    "totalResults": "1"
                }
            }
```

Business Logic	Comments
	} }

3. Retrieve Reservation Rate Break Down using getRateInfo The synchronous getRateInfo can be used to retrieve a detailed breakdown of reservation rates. This information includes rate codes, room types, and rate amounts, which are crucial for analyzing pricing strategies and ensuring accurate billing. The data can be utilized for revenue management, operational planning, and detailed reporting purposes.

Business Logic	Comments
Retrieving the Reservation Rate Breakdown Data	You can retrieve the reservation rate breakdown data based on the following parameters: • **id**: The unique identifier for the specific reservation or block being queried. • **ratePlanCode**: The code of the rate plan being queried. • **roomType**: The type of room associated with the rate plan. • **adults**: Number of adults included in the reservation. • **criteriaStartDate**: The start date for the rate criteria. • **criteriaEndDate**: The end date for the rate criteria. • **hotelId**: Specifies the hotel for which the rate information is being queried.
Key Response Data Elements	Rate Information includes: • **Rate Plan Code**: Identifier for the rate plan. • **Room Type**: Type of room the rate applies to. • **Rate Amount**: The price of the room per night. • **Rate Currency**: The currency in which the rate is denoted. • **Rate Description**: Description of the rate plan. • **Market Code**: Market segment associated with the rate. • **Source Code**: Source of the booking (for example, direct, OTA). • **Total Revenue**: Calculated total revenue based on the rate and duration. • **Tax Amount**: Applicable taxes on the rate. • **Net Rate Amount**: Rate amount after applying any discounts or adjustments. • **Daily Rate Details**: Breakdown of rates on a daily basis within the specified date range.

Business Logic	Comments

Sample API request/ response	getRateInfo
	
	Endpoint:
	GET {{HostName}}/rsv/v1/hotels/{{HotelId}}}/ reservations/rateInfo? ratePlanCode=BC&riteriaStartDate=2023-11-29&criteriaEndDate=2023-11-29
	
	Sample Successful (200 OK) Response:
	{
	"summary": {
	"details": [
	{
	"summaryDate": "2023-11-29",
	"revenue": 0,
	"package": 0,
	"tax": 0,
	"gross": 0,
	"net": 0,
	"ratePlanCode": "BC",
	"currencyCode": "NOK"
	}
	],
	"gross": 0,
	"net": 0,
	"currencyCode": "NOK",
	"start": "2023-11-29",
	"end": "2023-11-29",
	"hasSuppressedRate": false
	},
	"links": []
	}

4. Retrieve Reservation Stay Records using getStayHistory

The synchronous operation getStayHistory is designed to retrieve detailed records of a guest's stays at a hotel. This data provides a comprehensive overview of a guest's past visits, which is valuable for various operational and analytical purposes.

Business Logic	Comments
Retrieving the Reservation Stay Records	Once a guest checks out, you can retrieve the reservation stay records history based on the following parameters: • **profileId**: Fetches the stay history for the specific guest profile. • **hotelIds**: Filters the stay history to include only specified hotels. • **reportType**: Defines the type of report to generate (for example, ProfileStayRecords for stay history). • **stayFrom**: The beginning date to filter the stays. • **stayTo**: The ending date to filter the stays. • **summary**: Determines whether the response includes a summary (true) or detailed records (false).
Key Response Data Elements	Stay History Information Includes the following: Stay Records: • **Hotel ID**: Identifier for the hotel where the stay occurred. • **Arrival Date**: Date when the guest checked in. • **Departure Date**: Date when the guest checked out. • **Room Type**: Type of room the guest stayed in. • **Room Number**: Specific room number assigned to the guest. • **Rate Code**: Code representing the rate plan applied. • **Rate Amount**: The amount charged per night. • **Rate Currency**: Currency in which the rate is denoted. • **Total Revenue**: Total revenue generated from the stay. • **Room Revenue**: Revenue from the room charges. • **Food & Beverage Revenue**: Revenue from food and beverage purchases. • **Other Revenue**: Revenue from other services or products. • **Special Requests**: Any special requests made by the guest during their stay. • **Payment Method**: Method of payment used by the guest. • **Card Type**: Type of credit card used (if applicable). • **Last Four Digits**: Last four digits of the card used for payment. Guest Preferences and Comments: • **Preferences**: Specific preferences indicated by the guest. • **Comments**: Any additional comments or notes related to the stay.

Business Logic	Comments

Sample API request/ response	getStayHistory
	
	Endpoint:
	GET {{HostName}}/crm/v1/profiles/{{ProfileId}}/stayHistory?
	fetchInstructions=HistoryReservation
	
	Sample Successful (200 OK) Response:
	{
	"profileStayDetails": {
	"reservationInfoList": {
	"historyList": {
	"reservationInfo": [
	{
	"reservationIdList": [
	{
	"id": "36901",
	"type": "Reservation"
	},
	{
	"id": "3577366",
	"type": "Confirmation"
	}
	],
	"roomStay": {
	"arrivalDate": "2023-05-23",
	"departureDate": "2023-05-24",
	"roomType": "QE",
	"ratePlanCode": "48HRS",
	"rateAmount": {
	"amount": 2590,
	"currencyCode": "NOK"
	},
	"rateSuppressed": false,
	"guarantee": {
	"guaranteeCode": "GTDCCOPI"
	}
	},
	"attachedProfiles": [],
	"hotelId": "SH765",
	"reservationStatus": "NoShow",
	"computedReservationStatus": "NoShow"
	}
	],
	"hasMore": false,
	"totalResults": 1
	},
	"futureList": {
	"hasMore": false,
	"totalResults": 0
	}

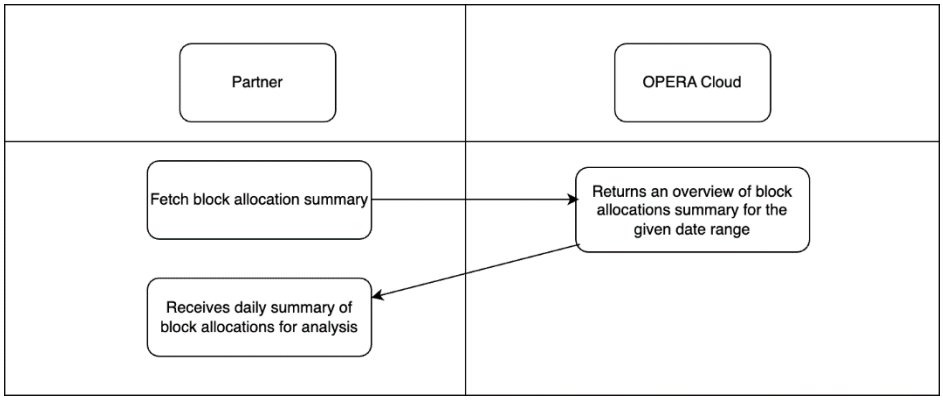
Business Logic	Comments

	<pre> }
	    },
	    "links": []
	}</pre>

Block Management

Block management involves the organization and administration of group reservations or blocks of rooms in a hotel. This process in OPERA Cloud includes setting aside rooms for events, conferences, or group bookings, ensuring availability, and managing associated logistics and billing. You can retrieve block allocations to manage group reservations and optimize room inventory. This data helps in dynamic pricing adjustments and accurate forecasting for better revenue management in OPERA Cloud.

Figure 4-3 Workflow



Use Cases

Use Case	Description	Operation
Retrieve details about your block allocations	You can fetch block allocation information for a hotel within a specified date range. This returns details about the allocated inventory, rates, and room type statistics, aiding in the management of group reservations and optimization of room inventory.	getBlockAllocationSummary

Retrieve Block Allocations Summary Data using getBlockAllocationSummary

The asynchronous operation getBlockAllocationSummary can be used to retrieve detailed information about room block allocations. This information includes the number of rooms allocated, block dates, and associated rates, which are crucial for managing group reservations and optimizing room inventory. The data can be utilized for dynamic pricing adjustments, accurate forecasting, and effective revenue management.

Business Logic	Comments
Retrieving the initial block allocations summary data using start/endDate	The block allocation summary data can be requested using startDate and endDate to fetch the initial block allocations summary data for a specified date range: • **startDate**: The beginning date to filter block allocations. • **endDate**: The ending date to filter block allocations. Other Critical Parameters include: • **hotelId**: Specifies the hotel for which block allocations are being queried. • **blockId**: Retrieves a specific block by its unique ID. • **status**: Filters blocks based on their current status. • **roomType**: Filters block allocations by the type of room. • **rateCode**: Filters block allocations based on the rate code applied.
Retrieving the incremental block allocations summary Data using startLastModifiedDate/ endLastModifiedDate	The block summary data can be requested using startLastModifiedDate/endLastModifiedDate to fetch the reservation summary data for a given date range: • **startLastModifiedDate**: Filters reservations that have been modified on or after this date. • **endLastModifiedDate**: Filters reservations that have been modified on or before this date.
Key Considerations	• The hotelId parameter is essential to identify which hotel's block allocations are being queried. • Keep the date range (startDate and endDate or startLastModifiedDate and endLastModifiedDate) as narrow as possible to reduce the search window to a specific time frame. • Filtering options like blockId, status, roomType, and rateCode allow for more precise and relevant search results, ensuring that the data retrieved is highly specific to your needs.
Key Response Data Elements	The key response data elements include the following: • Hotel Information: – **Hotel ID**: Identifier for the hotel. • Block Information: – **Block ID**: Unique identifier for the block of rooms. – **Status**: Current status of the block (for example, confirmed, tentative). • Allocation Information: – Allocation Dates: * **Start Date**: The beginning date of the block allocation. * **End Date**: The ending date of the block allocation. – **Room Type**: Description of the room type allocated in the block. – **Allocated Rooms**: Number of rooms allocated in the block. – **Available Rooms**: Number of rooms still available within the block. • Rate Information: – **Rate Code**: Code representing the rate plan applied to the block. – **Rate Amount**: The amount charged per room in the block. – **Rate Currency**: The currency in which the rate is denoted. • Modification Information: – **Last Modified Date**: The date and time when the block was last modified.

Business Logic	Comments

Sample API request/ response	1. startBlockAllocationSummaryProcess
	1.1. startBlockAllocationSummaryProcess (requested using the parameters startDate/endDate)
	startBlockAllocationSummaryProcess (startDate/endDate)
	
	Endpoint:
	POST {{HostName}}/blk/async/v1/externalSystems/{{ExtSystemCode}}/hotels/{{HotelId}}/blocks/allocationSummary
	
	Request Body:
	{
	"startLastModifiedDate": "2024-07-01",
	"endLastModifiedDate": "2024-07-01"
	}
	
	Sample Successful Response:
	202 Accepted
	
	1.2. startBlockAllocationSummaryProcess (requested using the parameters startLastModifiedDate/endLastModifiedDate)
	startBlockAllocationSummaryProcess (startLastModifiedDate/ endLastModifiedDate)
	
	Endpoint:
	POST {{HostName}}/blk/async/v1/externalSystems/{{ExtSystemCode}}/hotels/{{HotelId}}/blocks/allocationSummary
	
	Sample Successful Response:
	202 Accepted
	
	2. getBlockAllocationSummaryProcessStatus
	getBlockAllocationSummaryProcessStatus
	
	Endpoint:
	HEAD {{HostName}}/blk/async/v1/externalSystems/{{ExtSystemCode}}/hotels/{{HotelId}}/blocks/allocationSummary/{{RequestId1}}
	
	Sample Request cURL:
	
	curl --location --head '{{HostName}}/blk/async/v1/externalSystems/{{ExtSystemCode}}/hotels/{{HotelId}}/blocks/allocationSummary/{{RequestId1}}'
	--header 'Content-Type: application/json' \
	--header 'x-hotelId: {{HotelId}}' \
	--header 'x-app-key: {{AppKey}}' \
	--header 'Authorization: Bearer {{Token}}' \

Business Logic	Comments

	--data ''
	
	Sample Successful (200 OK) Response:
	
	201 Created
	(You are ready to proceed to Step 3)
	
	3. getBlockAllocationSummary
	getBlockAllocationSummary
	
	Endpoint:
	GET {{HostName}}/blk/async/v1/externalSystems/
	{{ExtSystemCode}}/hotels/{{HotelId}}/blocks/
	allocationSummary/{{RequestId2}}
	
	Sample Successful (200 OK) Response:
	[
	{
	"blockId": 1825245,
	"blockCode": "TOU280723_007",
	"blockName": "touroomgrid",
	"masterBlockId": 1825246,
	"status": "ACT",
	"marketCode": "IPL",
	"startDate": "2023-07-28T00:00:00",
	"endDate": "2023-07-29T00:00:00",
	"cutOffDays": 0,
	"cateringOwner": 12213,
	"cateringOwnerCode": "ALL",
	"owner": 12213,
	"ownerCode": "ALL",
	"roomsOwner": 12213,
	"roomsOwnerCode": "ALL",
	"createDateTime": "2023-07-28T05:12:59",
	"blockType": "G",
	"lastModifiedDate": "2023-07-28T05:12:59",
	"sourceCode": "WI",
	"bookingMedium": "WI",
	"allocationDates": [
	{
	"allocationDate":
	"2023-07-28T00:00:00",
	"allocations": [
	{
	"originalRooms": 10,
	"roomType": "TR",
	"currentRooms": 0,
	"pickupRooms": 0,
	"inventory": {

Business Logic	Comments

	"onePerson": 0,
	"twoPerson": 0,
	"threePerson": 0,
	"fourPerson": 0,
	"sellLimit": 0
	},
	"rates": {
	"onePerson": 500,
	"twoPerson": 0,
	"threePerson": 0,
	"fourPerson": 0,
	"extraPerson": 0
	},
	"actualRevenue": {
	"currency": "NOK"
	}
	},
	{
	"originalRooms": 10,
	"roomType": "QE",
	"currentRooms": 0,
	"pickupRooms": 0,
	"inventory": {
	"onePerson": 0,
	"twoPerson": 0,
	"threePerson": 0,
	"fourPerson": 0,
	"sellLimit": 0
	},
	"rates": {
	"onePerson": 500,
	"twoPerson": 0,
	"threePerson": 0,
	"fourPerson": 0,
	"extraPerson": 0
	},
	"actualRevenue": {
	"currency": "NOK"
	}
	},
	{
	"originalRooms": 10,
	"roomType": "SR",
	"currentRooms": 0,
	"pickupRooms": 0,
	"inventory": {
	"onePerson": 0,
	"twoPerson": 0,
	"threePerson": 0,
	"fourPerson": 0,
	"sellLimit": 0

Business Logic	Comments

	<pre> },
	                        "rates": {
	                            "onePerson": 500,
	                            "twoPerson": 0,
	                            "threePerson": 0,
	                            "fourPerson": 0,
	                            "extraPerson": 0
	                        },
	                        "actualRevenue": {
	                            "currency": "NOK"
	                        }
	                    }
	                ]
	            }
	        ]
	    }
	]</pre>

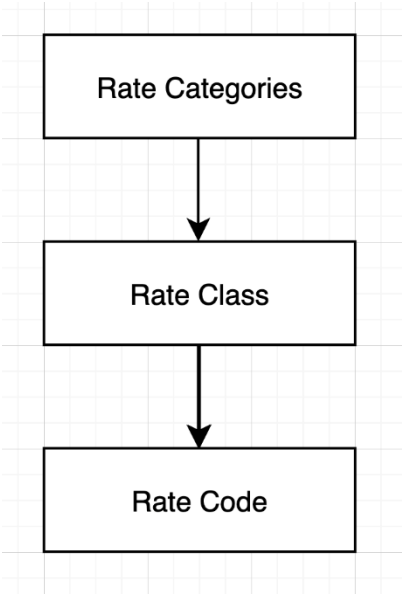
Rate Management

Rate management in OPERA Cloud involves setting and adjusting room rates to maximize revenue and occupancy. This includes creating rate plans, dynamic pricing adjustments, and retrieving best available rates and hurdle rates for optimized pricing.

Rate Structure in OPERA Cloud

OPERA Cloud rate structure is divided into Rate Class, Rate Category, and Rate Code. Rate categories are high-level classifications that group various rate codes into broader categories. They help in organizing and managing rates more effectively. Examples of Rate Categories could be Corporate, Leisure, and so on. Rate classes are more granular and specific classifications within the rate structure. They can be used to define specific characteristics or attributes of rates within a category. Examples of rate classes could be Negotiate, Wholesale, Discounted, and so on. You must define one or more rate classes and a rate class is then associated to each Rate Category. Rate classes can be used to query rates from OPERA Cloud. For more information on the complex Rate Structures in OPERA Cloud, refer to Configuring Rate Classes in the OPERA Cloud User Guide.

Figure 4-4 Rate Structure in OPERA Cloud



Hurdles in OPERA Cloud

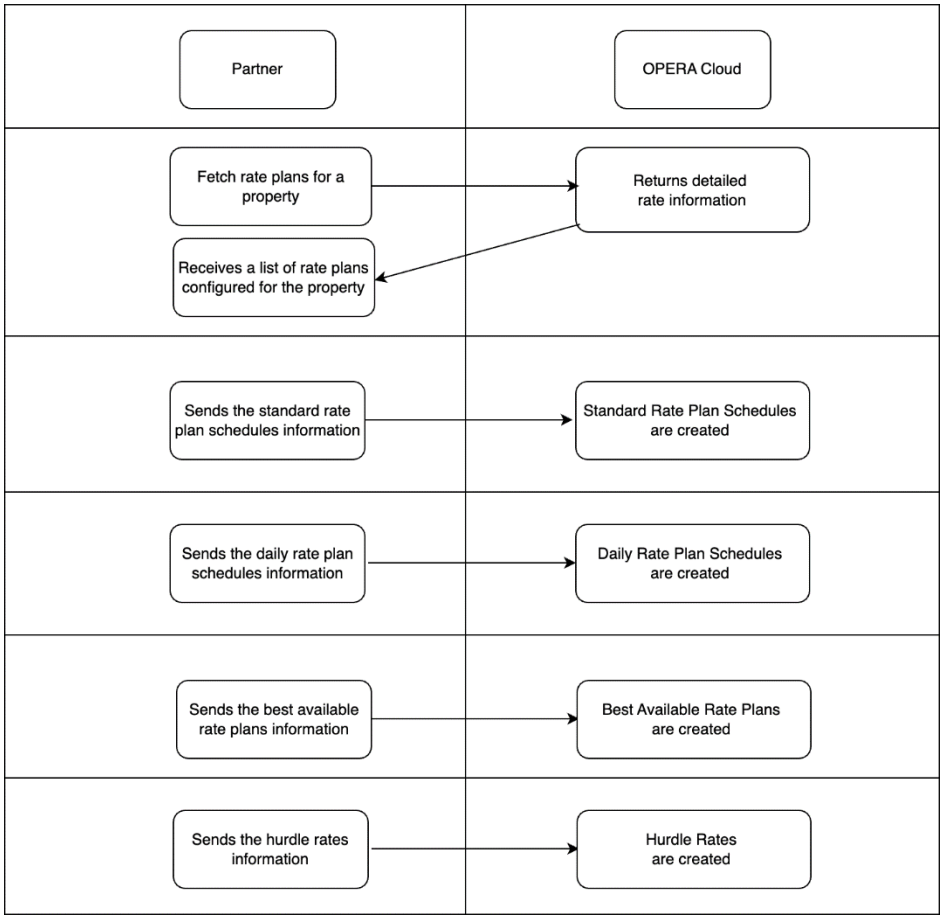
Hurdle Rates represent the monetary hurdle value calculated by the revenue management solution after considering the information received from OPERA Cloud. This is the value that must be reached for OPERA Cloud to display a rate code/room type on the rate grid. For more information, refer to the Hurdle Rates topic in the OPERA Cloud User Guide.

Use Cases

Use Case	Description	Operation
Retrieving Rate Plans for a property	You can retrieve a list of rate plans for a specified property, including options to filter inactive rate plans and include detailed rate information.	getRatePlans
Creating/Updating the Standard Rate Plan Schedules	You can add and or/update standard rate plans for a specified property. You can add new standard rate plan schedules, including detailed rate information and options to define specific attributes for each schedule. By dynamically adjusting room rates based on factors like demand, occupancy, and market conditions, rate plan schedules help optimize revenue and ensure competitive pricing.	postRatePlanSchedules
Creating/Updating the Daily Rate Plan Schedules	You can add and/or update the rate price schedule to existing OPERA Daily Rate plans. Rate Plan Schedules can dynamically adjust room rates based on various factors, such as demand, occupancy, and market conditions. This helps in optimizing revenue and maintaining competitive pricing.	startSetDailyRatePlanSchedulesProcess

Use Case	Description	Operation
Creating/Updating the Best Available Rate Plans	You can create/update the Best Available Rate Plans by Length of Stay or by Day by posting them to OPERA Cloud. This helps in dynamically offering the most competitive and optimized pricing to guests based on their intended stay duration.	startSetBestAvailableRatePlansProcess
Creating Hurdle Rates	Hurdle Rates represent the monetary hurdle value calculated by the revenue management solution after considering the information received from OPERA Cloud. You can create Hurdle Rates by posting them to OPERA Cloud. This can help in managing revenue by setting pricing thresholds that prevent underselling during high demand periods.	startHurdleRatesProcess

Figure 4-5 Rate Management Workflow



1. Retrieving Rate Plans for a property

The synchronous operation getRatePlans in OPERA Cloud allows users to retrieve a list of rate plans for a specified property. This operation includes options to filter the rate plans based on active/inactive status and to include rate information if required. This is essential for

revenue management systems (RMS) to analyze and manage different pricing strategies and packages offered by the hotel.

Business Logic	Comments
Retrieving Rate Plans for a property	You can retrieve the rate plans information based on the following parameters: • **hotelId**: Specifies the hotel for which rate plans are being queried. • **includeRateInformation**: Boolean flag to specify whether to include detailed rate information in the response. • **includeInactive**: Boolean flag to specify whether to include inactive rate plans in the response. • **offset**: An integer parameter to specify the offset for pagination.
Key response data elements	The key response data elements include: • **ratePlanCode**: Code identifying the specific rate plan. • **description**: Description of the rate plan. • **startDate**: The date from which the rate plan is applicable. • **endDate**: The date until which the rate plan is applicable. • **status**: Status of the rate plan (that is, Active, Inactive). • **rateDetails**: Detailed rate information is returned if the includeRateInformation is set to true which includes the following: – **onePersonRate**: Rate for one person. – **twoPersonRate**: Rate for two persons. – **threePersonRate**: Rate for three persons. – **fourPersonRate**: Rate for four persons. – **extraPersonRate**: Rate for each additional person. – **extraChildRate**: Rate for each additional child. – **rateByAgeBuckets**: Rates applicable based on the age of guests, provided as an array. – **minimumChildrenForFreeStay**: Minimum number of children allowed to stay for free. – **pointsRequired**: Points required for the rate plan if applicable. – **overrideFloorAmount**: Boolean indicating if the floor amount should be overridden.

Business Logic	Comments

Sample API request/ response	getRatePlans
	
	Endpoint:
	
	GET {{HostName}}/rtp/v1/ratePlans?
	includeRateInformation=false&includeInactive=true&off
	set=0&sellDate=2023-06-03&hotelId={{HotelId}}Sample
	Successful (200 OK) Response:
	{
	"ratePlanShortInfoList": {
	"ratePlanShortInfo": [
	{
	"primaryDetails": {
	"description": {
	"defaultText": "Corporate
	Saver BAR"
	},
	"startSellDate": "2018-10-10",
	"endSellDate": "2049-12-31",
	"sellSequence": 361,
	"privilegedRate": false,
	"privilegedRateRestriction":
	false,
	"lockStatus": "Unlocked"
	},
	"classifications": {
	"rateCategory": "BAR",
	"marketCode": "CV"
	},
	"hotelId": "SH765",
	"ratePlanCode": "NCS6B",
	"discountAllowed": true
	}
	],
	"hasMore": true,
	"totalResults": 1,
	"offset": 20,
	"limit": 20,
	"totalPages":
	},
	"masterInfoList": [
	{
	"codeInfo": [
	{
	"description": "Special Rates",
	"code": "SPECIAL"
	}
	],
	"codeType": "DisplaySet"
	}

Business Logic	Comments
	] }

2. Creating/Updating the Standard Rate Plan Schedules using postRatePlanSchedules

Standard Pricing Schedules in OPERA Cloud enable you to manage daily rates for each room type on a daily basis. You can also select one or more package items for a pricing schedule if it is required for offering a special entitlement for specific room types / date ranges only. For example, you might wish to include Champagne in the rate only for the Executive Suite; the package item is then added at the pricing schedule level and not for the entire rate code (all room types).

The synchronous operation postRatePlanSchedules is used to set or update standard daily rate plan schedules for a specified date range. This information includes rate amounts for different occupancy levels, room types, and special conditions.

Business Logic	Comments
Posting the standard rate plan schedules	You can set or update the standard rate plan schedules using specific parameters for a given date range: • **hotelId**: Specifies the hotel for which rate plans are being set. • **ratePlanCode**: Code identifying the specific rate plan. • **rateDate**: The date for which the rate is applicable. • **roomType**: The type of room to which the rate applies. The rate plan schedule can be posted with the following details: • **onePersonRate**: Rate for one person. • **twoPersonRate**: Rate for two persons. • **threePersonRate**: Rate for three persons. • **fourPersonRate**: Rate for four persons. • **fivePersonRate**: Rate for five persons. • **extraPersonRate**: Rate for each additional person. • **extraChildRate**: Rate for each additional child. • **oneChildRate**: Rate for one child. • **twoChildrenRate**: Rate for two children. • **threeChildrenRate**: Rate for three children. • **fourChildrenRate**: Rate for four children. • **rateByAgeBuckets**: Rates applicable based on the age of guests. • **minimumChildrenForFreeStay**: Minimum number of children allowed to stay for free. • **pointsRequired**: Points required for the rate plan. • **overrideFloorAmount**: Boolean indicating if the floor amount should be overridden.

Business Logic	Comments
Key Considerations	• The hotelId parameter is crucial to specify which hotel's rate plans are being set or updated. • Clearly define the rateDate for which the rates are applicable to avoid overlaps and gaps in the rate plans. • Use accurate ratePlanCode and roomType to ensure the rates are applied correctly. • Clearly specify rate amounts for different occupancy levels (for example, onePersonRate, twoPersonRate) and additional charges (for example, extraPersonRate, extraChildRate). • Use the overrideFloorAmount flag carefully to prevent unintended rate overrides. • Set appropriate rateByAgeBuckets to cater to different age groups, ensuring correct rate applications for children and additional guests. • Define the minimumChildrenForFreeStay accurately to align with promotional offers and policies. • Include pointsRequired for loyalty programs to maintain consistency with guest rewards.
Validation	Validate the applied standard rate plan schedules through the RMS to ensure they match the intended configurations.

Business Logic	Comments

Sample API request/ response	postRatePlansSchedules
	
	Endpoint:
	
	GET {{HostName}}/rtp/v1/ratePlans?includeRateInformation=false&includeInactive=true&offset=0&sellDate=2023-06-03&hotelId={{HotelId}}Endpoint:
	POST {{HostName}}/rtp/v1/hotels/{{HotelId}}/ratePlans
	
	Request Body:
	{
	"ratePlan": {
	"primaryDetails": {
	"description": {
	"defaultText": "Rate Code Description Goes Here test"
	},
	"startSellDate": "2023-05-03",
	"endSellDate": "2023-12-31",
	"privilegedRate": false,
	"privilegedRateRestriction": false,
	"lockStatus": "Unlocked"
	},
	"classifications": {
	"rateCategory": "HOTEL",
	"marketCode": "LEISURE"
	},
	"transactionDetails": {
	"folioText": {
	"defaultText": "Accommodation"
	},
	"transactionCode": "1000",
	"currencyCode": "USD",
	"rateIncludesTax": true
	},
	"rateCommission": {},
	"roomTypeList": [
	{
	"code": "ECO"
	},
	{
	"code": "SGL"
	},
	{
	"code": "DBL"
	},
	{
	"code": "SUP"
	},
	{

Business Logic	Comments

	"code": "JSUI"
	},
	{
	"code": "SUI"
	}
	],
	"ratePackages": {
	"packages": [],
	"packageGroups": []
	},
	"controls": {
	"sell": {
	"minimumStayThrough": "1",
	"minimumLengthOfStay": "1"
	},
	"yield": {
	"yieldable": "NonYieldable"
	}
	},
	"distribution": {
	"restrictionUpdate": false,
	"rateUpdate": false,
	"myFidelioUploadAllowed": false,
	"channelRateMappingExists": false,
	"channelAllowed": false
	},
	"ratePlanBasedOnRates": [
	{
	"basedOnRatePlanType": "None"
	}
	],
	"approvalNotes": [
	{}
	],
	"mobileCheckinAllowed": false,
	"mobileCheckoutAllowed": false,
	"hotelId": "{{HotelId}}",
	"ratePlanCode": "TEST99",
	"supressRate": false,
	"printRate": true,
	"discountAllowed": false,
	"redemption": false,
	"bARRate": false,
	"daily": false,
	"tiered": false,
	"dayUse": false,
	"dayType": false,
	"complimentary": false,
	"houseUse": false,
	"negotiated": false,
	"ownerRate": false,

Business Logic	Comments

	<pre>"membershipEligible": true,
	"advancedDailyBase": false,
	"advancedDailyRate": false
	}
	}
	
	Sample Successful Response:
	
	201 Created<The Standard Rate Plan Schedules are posted in OPERA Cloud></pre>

3. Creating/Updating the Daily Rate Plan Schedules using startSetDailyRatePlanSchedulesProcess

Daily rates are identified by the Rate Category of Daily. The asynchronous operation startSetDailyRatePlanSchedulesProcess is used to set or update daily rate plan schedules for a specified date range. This information includes rate amounts for different occupancy levels, room types, and special conditions. The data is essential for implementing dynamic pricing strategies, ensuring competitive rates, and optimizing revenue. It can be utilized for operational planning, revenue forecasting, and detailed rate management reporting.

Business Logic	Comments
Posting the daily rate plan schedules	The daily rate plan schedules can be requested using specific parameters to set or update the rate plan schedules for a specified date range: • **hotelId**: Specifies the hotel for which rate plans are being set. • **ratePlanCode**: Code identifying the specific rate plan. • **rateDate**: The date for which the rate is applicable. • **roomType**: The type of room to which the rate applies. You can send the specific rate plan schedule to be posted with the following information: • **onePersonRate**: Rate for one person. • **twoPersonRate**: Rate for two persons. • **threePersonRate**: Rate for three persons. • **fourPersonRate**: Rate for four persons. • **fivePersonRate**: Rate for five persons. • **extraPersonRate**: Rate for each additional person. • **extraChildRate**: Rate for each additional child. • **oneChildRate**: Rate for one child. • **twoChildrenRate**: Rate for two children. • **threeChildrenRate**: Rate for three children. • **fourChildrenRate**: Rate for four children. • **rateByAgeBuckets**: Rates applicable based on the age of guests. • **minimumChildrenForFreeStay**: Minimum number of children allowed to stay for free. • **pointsRequired**: Points required for the rate plan. • **overrideFloorAmount**: Boolean to indicate if the floor amount should be overridden.

Business Logic	Comments
Key Considerations	• The hotelId parameter is essential to identify which hotel's rate plans are being set or updated. • Specify the exact rateDate for which the rates are applicable to avoid overlaps and gaps in the rate plans. • Use precise ratePlanCode and roomType to ensure the rates are applied correctly. • Clearly define rate amounts for different occupancy levels (for example, onePersonRate, twoPersonRate) and additional charges (for example, extraPersonRate, extraChildRate). • Use the overrideFloorAmount flag judiciously to prevent unintended rate overrides. • Set appropriate rateByAgeBuckets to cater to different age groups, ensuring correct rate applications for children and additional guests. • Define the minimumChildrenForFreeStay accurately to align with promotional offers and policies. • Include pointsRequired for loyalty programs to maintain consistency with guest rewards.
Validation	Validate the applied daily rate plan schedules through the RMS to ensure they match the intended configurations.

Business Logic	Comments

Sample API request/ response	1. startSetDailyRatePlanSchedulesProcess
	
	Endpoint:
	
	POST {{HostName}}/rtp/async/v1/externalSystems/ {{ExtSystemCode}}/hotels/{{HotelId}}/ratePlans/ dailySchedules
	
	Request Body:
	
	{
	"dailyRatePlanSchedule": [
	{
	"rateAmounts": {
	"onePersonRate": 1005,
	"twoPersonRate": 1500,
	"threePersonRate": 0,
	"fourPersonRate": 0,
	"fivePersonRate": 0,
	"extraPersonRate": 0,
	"extraChildRate": 0,
	"oneChildRate": 0,
	"twoChildrenRate": 0,
	"threeChildrenRate": 0,
	"fourChildrenRate": 0,
	"rateByAgeBuckets": [
	{
	"rateAmount": 0,
	"minimumAge": 0,
	"maximumAge": 0
	}
	],
	"minimumChildrenForFreeStay": 0,
	"pointsRequired": 0,
	"overrideFloorAmount": true
	},
	"rateDate": "2023-11-07",
	"ratePlanCode": "ADELR",
	"roomType": "QE"
	}
	]
	}
	
	Sample Successful Response:
	
	202 Accepted

Business Logic	Comments

	2. getDailyRatePlanSchedulesProcessStatus
	
	`HEAD {{HostName}}/rtp/async/v1/externalSystems/{{ExtSystemCode}}/hotels/{{HotelId}}/ratePlans/dailySchedules/{{RequestId1}}`
	
	`Sample Successful Response:`
	
	`201 Created`
	`(You are ready to proceed to Step 3)`
	
	3. getDailyRatePlanSchedules
	
	`Endpoint:`
	
	`GET {{HostName}}/rtp/async/v1/externalSystems/{{ExtSystemCode}}/hotels/{{HotelId}}/ratePlans/dailySchedules/{{RequestId2}}`
	
	`Sample Successful Response:`
	
	`200 OK`
	`<The Daily Rate Plan Schedules are posted to OPERA Cloud.>`

4. Creating/Updating the Best Available Rate Plans using startSetBestAvailableRatePlansProcess

The asynchronous operation startSetBestAvailableRatePlansProcess can be used to set or update the best available rate plans for specified lengths of stay (LOS). This information includes rate plan codes and applicable dates, which are crucial for offering competitive and optimized pricing to guests. The data can be utilized for dynamic pricing adjustments, enhancing revenue management strategies, and ensuring that the most attractive rates are available for different stay durations.

Business Logic	Comments
Posting the best available rate plans	The best available rate plans can be set using specific parameters to determine the most competitive rates for specified lengths of stay (LOS): • **hotelId**: Specifies the hotel for which rate plans are being set. You can send the specific rate plan schedule to be posted with the following information: • **ratePlanCodes**: An array of rate plan codes to be considered for the best available rate. • **rateDate**: The date for which the best available rate is to be determined. • **lOS1 to lOS8**: Flags indicating the length of stay restrictions for 1 to 8 nights.

Business Logic	Comments
Key Considerations	• The hotelId parameter is essential to identify which hotel's best available rate plans are being set or updated. • Specify the exact rateDate for which the best available rates are to be determined to avoid overlaps and gaps in the rate plans. • Include relevant ratePlanCodes to ensure the rates are derived from the correct rate plans, providing accurate and competitive pricing. • Set the LOS flags (los1 to los8) judiciously to ensure the best available rates are fetched for the intended lengths of stay, avoiding unnecessary data retrieval. • Ensure rate plan codes in the ratePlanCodes array are valid and currently active to prevent errors in rate retrieval.
Validation	Validate the applied best available rate plans through the RMS to ensure they match the intended configurations.

Business Logic	Comments

Sample API request/ response	1. startSetBestAvailableRatePlansProcess
	startSetBestAvailableRatePlansProcess
	
	Endpoint:
	
	POST {{HostName}}/rtp/async/v1/externalSystems/ {{ExtSystemCode}}/hotels/{{HotelId}}/ratePlans/ bestAvailableLOS
	
	Sample Request cURL:
	{
	"bestAvailableRatePlans": [
	{
	"ratePlanCodes": [
	"string"
	],
	"rateDate": "2023-11-10",
	"los1": true,
	"los2": false,
	"los3": false,
	"los4": false,
	"los5": false,
	"los6": false,
	"los7": false,
	"los8": false
	}
	]
	}
	
	Sample Successful Response:
	
	202 Accepted
	
	2. getBestAvailableRatePlansProcessStatus
	
	Endpoint:
	
	HEAD {{HostName}}/rtp/async/v1/externalSystems/ {{ExtSystemCode}}/hotels/{{HotelId}}/ratePlans/ bestAvailableLOS/{{RequestId1}}
	
	Sample Request cURL:
	
	curl --location '{{HostName}}/rtp/async/v1/ externalSystems/{{ExtSystemCode}}/hotels/{{HotelId}}/ ratePlans/bestAvailableLOS/{{RequestId1}}' \

Business Logic	Comments
	--header 'Content-Type: application/json' \ --header 'x-hotelId: {{HotelId}}' \ --header 'x-app-key: {{AppKey}}' \ --header 'Authorization: Bearer {{Token}}' \ --data '' Sample Successful Response: 201 Created (You are ready to proceed to Step 3) 3. getBestAvailableRatePlans Endpoint: GET {{HostName}}/rtp/async/v1/externalSystems/ {{ExtSystemCode}}/hotels/{{HotelId}}/ratePlans/ bestAvailableLOS/{{RequestId2}} Sample Request cURL: curl --location '{{HostName}}/rtp/async/v1/ externalSystems/{{ExtSystemCode}}/hotels/{{HotelId}}/ ratePlans/bestAvailableLOS/{{RequestId2}}' \ --header 'Content-Type: application/json' \ --header 'x-hotelId: {{HotelId}}' \ --header 'x-app-key: {{AppKey}}' \ --header 'Authorization: Bearer {{Token}}' \ --data '' Sample Successful Response: 200 OK <The Best Available Rate Plans are posted to OPERA Cloud.>

5. Creating Hurdle Rates using startHurdleRatesProcess

The asynchronous operation startHurdleRatesProcess is used to set or update hurdle rates for a property. This information is crucial for maintaining pricing integrity, maximizing revenue, and strategically managing room availability. The data can be utilized for revenue management, demand forecasting, and operational planning.

	Comments
Posting the hurdle rates	The hurdle rates can be set using specific parameters to establish the minimum acceptable rates for room bookings: • **hotelId**: Specifies the hotel for which hurdle rates are being set. You can send the specific hurdle rate details to be posted with the following information: • **hurdleDate**: The date for which the hurdle rate is to be applied. • **roomType**: The type of room to which the hurdle rate applies. • **roomCategory**: The category of the room. • **yieldCategory**: The yield category for revenue management. • **lengthOfStay**: The length of stay for which the hurdle rate applies. • **yieldMarketCode**: The market code used for yield management. • **hurdle**: The minimum rate that can be accepted. • **delta**: The difference amount to adjust the hurdle rate. • **ceiling**: The maximum rate that can be charged. • **maximumSolds**: The maximum number of rooms to be sold at this rate. • **roomsSold**: The number of rooms already sold at this rate. • **override**: A flag indicating if the hurdle rate should override existing rates.
Key Considerations	• Specify the exact hurdleDate for which the hurdle rates are to be applied to avoid overlaps and gaps in the rate plans. • Include relevant roomType and roomCategory to ensure the rates are applied to the correct room configurations, providing accurate pricing. • Set the lengthOfStay parameter accurately to ensure hurdle rates are applied correctly for the intended durations of stay. • Use yieldCategory and yieldMarketCode appropriately to align with the hotel's revenue management strategy and market segmentation. • Clearly define the hurdle and ceiling amounts to establish minimum and maximum acceptable rates, optimizing revenue. • Accurately set the maximumSolds and roomsSold parameters to manage inventory and ensure proper tracking of sold rooms. • Use the override flag judiciously to determine whether the hurdle rate should override existing rates, preventing unintended rate changes.
Validation	Validate the applied hurdle rates through the RMS to ensure they align with the intended configurations and revenue strategies.

	Comments

Sample API request/ response	1. startHurdleRatesProcess
	startHurdleRatesProcess
	
	Endpoint:
	
	POST {{HostName}}/rtp/async/v1/externalSystems/ {{ExtSystemCode}}/hotels/{{HotelId}}/rates/hurdles
	
	Sample Request cURL:
	
	curl --location '{{HostName}}/rtp/async/v1/ externalSystems/{{ExtSystemCode}}/hotels/{{HotelId}}/ rates/hurdles' \
	--header 'Content-Type: application/json' \
	--header 'x-hotelId: {{HotelId}}' \
	--header 'x-app-key: {{AppKey}}' \
	--header 'Authorization: Bearer {{Token}}' \
	--data '[
	{
	"hurdleDate": "2023-11-11",
	"roomType": "QE",
	"roomCategory": "S",
	"yieldCategory": "AEL",
	"lengthOfStay": 0,
	"yieldMarketCode": "COMP",
	"hurdle": 0,
	"delta": 0,
	"ceiling": 0,
	"maximumSolds": 0,
	"roomsSold": 0,
	"override": true
	}
	]'
	
	Sample Successful Response:
	
	202 Accepted
	
	2. getHurdleRatesProcessStatus
	getHurdleRatesProcessStatus
	
	Endpoint:
	
	HEAD {{HostName}}/rtp/async/v1/externalSystems/ {{ExtSystemCode}}/hotels/{{HotelId}}/rates/hurdles/ {{RequestId1}}
	
	Sample Request cURL:
	Expand source

	Comments
	Sample Successful Response: 201 Created (You are ready to proceed to Step 3) 3. getHurdleRates getHurdleRates Endpoint: GET {{HostName}}/rtp/async/v1/externalSystems/{{ExtSystemCode}}/hotels/{{HotelId}}/rates/hurdles/{{RequestId2}} Sample Request cURL: curl --location '{{HostName}}/rtp/async/v1/externalSystems/{{ExtSystemCode}}/hotels/{{HotelId}}/rates/hurdles/{{RequestId2}}' \ --header 'Content-Type: application/json' \ --header 'x-hotelId: {{HotelId}}' \ --header 'x-app-key: {{AppKey}}' \ --header 'Authorization: Bearer {{Token}}' \ --data '' Sample Successful Response: 200 OK <The Best Available Rate Plans are posted to OPERA Cloud.>

Inventory Management

Inventory management in OPERA Cloud involves retrieving and analyzing inventory data for a hotel, including fetching inventory statistics for a specified date range and revenue inventory statistics for past, present, and future reservations. It also includes creating and posting sell limits to OPERA Cloud, ensuring optimal room availability and maximizing revenue through strategic inventory controls.

Figure 4-6 Inventory Management Workflow

Sell Limits in OPERA Cloud

Sell Limits let you control room availability at your property and change the inventory limits (oversell or undersell) at a property for given date/room type. You can configure sell limits from the Property Availability screen in the Inventory section of the OPERA Cloud application. For more information, refer to the Sell Limits topic in the OPERA Cloud Distribution User Guide.

Use Cases

Use Case	Description	Operation
Retrieve Inventory Statistics	You can retrieve inventory statistics for a specific property for a given date range. These statistics are crucial for revenue management systems (RMS) to analyze past performance, monitor current trends, and forecast future inventory needs.	getInventoryStatistics (synchronous)
Retrieving the Revenue Inventory Statistics	You can retrieve the revenue inventory statistics for the past, present, and future reservations from OPERA Cloud. You will be able to filter using stay date (with a start and end date) to fetch inventory data. You can group the data either by Room Type, Market Code, or Guarantee Code (Reservation Type) or you can gather data per date for the entire property.	getRevenueInventoryStatistics (asynchronous)
Creating Sell Limits	You can set the sell limits for a hotel by posting the data to OPERA Cloud. You will be able to specify the date range for which the sell limits apply and define limits either for the entire property by room type or by room class. This ensures precise control over room inventory, helping to optimize availability and maximize revenue.	postSellLimitsProcess

1. Retrieving Inventory Statistics using getInventoryStatistics

The synchronous operation getInventoryStatistics can be used to retrieve comprehensive inventory statistics for a property in a specified date range. The data can include various metrics related to room availability, sell limits, and occupancy rates. This is essential for managing room availability, occupancy, and optimizing revenue management strategies.

Business Logic	Comments
Retrieving the Inventory Statistics	The inventory statistics can be retrieved using specific parameters to gather data for a specified date range. **hotelId**: Specifies the hotel for which inventory statistics are being retrieved. You can send the specific inventory details to be fetched with the following parameters: • **dateRangeStart**: The starting date for the inventory statistics period. • **dateRangeEnd**: The ending date for the inventory statistics period. • **groupBy**: Specifies how to group the data, such as by Room Type, Market Code, or Guarantee Code (Reservation Type). • **roomType**: Filters the inventory data by specific room types. • **marketCode**: Filters the inventory data by market codes. • **guaranteeCode**: Filters the inventory data by guarantee codes. • **propertyLevel**: Boolean to indicate if the data should be gathered for the entire property.
Key Considerations	• The hotelId parameter is essential to identify which hotel's inventory statistics are being retrieved. • Specify the exact dateRangeStart and dateRangeEnd to ensure accurate and comprehensive data retrieval. • Use the groupBy parameter to organize data meaningfully (for example, by Room Type, Market Code, Guarantee Code) for detailed analysis. • Filter by roomType, marketCode, and guaranteeCode to obtain specific insights, helping tailor inventory strategies. • Set the propertyLevel flag accurately to decide if the data should be aggregated for the entire property or filtered by specific criteria. • Validate the retrieved inventory statistics through the RMS to ensure they align with the intended operational and revenue strategies.
Key Response Data Elements	The key response data elements include: • **hotelId**: The ID of the hotel for which inventory statistics are retrieved. • **dateRangeStart**: The starting date of the inventory statistics period. • **dateRangeEnd**: The ending date of the inventory statistics period. • **groupBy**: Specifies how the data is grouped for example, Room Type, Market Code, Guarantee Code). • **inventoryCounts**: Detailed counts of inventory for each grouping: – **availableRooms**: Number of rooms available. – **bookedRooms**: Number of rooms booked. – **outOfOrderRooms**: Number of rooms out of order.

Business Logic	Comments

Sample API request/ response	getInventoryStatistics
	
	Endpoint:
	
	{{HostName}}/inv/v1/hotels/{{HotelId}}/ inventoryStatistics
	
	Sample Successful (200 OK) Response:
	{
	"getInventoryStatistics":{
	"response":[
	{
	"statistics":[
	{
	"statisticDate":[
	{
	"statisticDate":"2020-01-25",
	"weekendDate":false
	},
	{
	"statisticDate":"2020-01-26",
	"weekendDate":false
	}
	],
	"statCode":"HOTEL1",
	"statCategoryCode":"HotelCode"
	},
	{
	"statisticDate":[
	{
	"inventory":[
	{
	"value":"1",
	"code":"SequenceId"
	}
	],
	"statisticDate":"2020-01-25",
	"weekendDate":false
	},
	{
	"inventory":[
	{
	"value":"1",
	"code":"SequenceId"
	}
	],
	"statisticDate":"2020-01-26",
	"weekendDate":false
	}
	],

Business Logic	Comments

	"statCode":"CSTND",
	"statCategoryCode":"HotelRoomCode",
	"statCodeClass":"STDCOMP",
	"description":"Component Room - Standard"
	},
	{
	"statisticDate":[
	{
	"statisticDate":"2020-01-25",
	"weekendDate":false
	},
	{
	"statisticDate":"2020-01-26",
	"weekendDate":false
	}
	],
	"statCode":"STD",
	"statCategoryCode":"HotelRoomCode",
	"statCodeClass":"ALL",
	"description":"Standard Room"
	}
	],
	"hotelName":"HOTEL1",
	"reportCode":"RoomsAvailabilitySummary"
	}
	]
	}
	}

2. Retrieving the Revenue Inventory Statistics using getRevenueInventoryStatistics

The asynchronous operation getRevenueInventoryStatistics endpoint retrieves detailed revenue inventory data for a property, which includes past, present, and future reservations. This data helps analyze revenue trends, manage inventory effectively, and make informed strategic decisions for optimizing room availability and pricing.

Business Logic	Comments
Retrieving the Revenue Inventory Statistics	The revenue inventory statistics can be retrieved using specific parameters to gather data for a specified date range: **hotelId**: Specifies the hotel for which revenue inventory statistics are being retrieved. You can send the specific revenue inventory details to be fetched with the following parameters: • **dateRangeStart**: The starting date for the revenue inventory statistics period. • **dateRangeEnd**: The ending date for the revenue inventory statistics period. • **roomType**: Filters the revenue inventory data by specific room types. • **marketCode**: Filters the revenue inventory data by market codes. • **guaranteeCode**: Filters the revenue inventory data by guarantee codes. • **propertyLevel**: Boolean to indicate if the data should be gathered for the entire property.
Key Considerations	• The hotelId parameter is crucial to ensure that statistics are retrieved for the correct hotel. • Specify accurate dateRangeStart and dateRangeEnd to cover the relevant period for analysis. • Filter by roomType, marketCode, and guaranteeCode to get specific revenue insights based on room types, market segments, and reservation guarantees. • Set the propertyLevel flag appropriately to determine whether to aggregate data at the property level or by specific criteria.
Key Response Data Elements	The key response data elements include: • **hotelId**: The ID of the hotel for which revenue inventory statistics are retrieved. • **dateRangeStart**: The start date of the period for which statistics are reported. • **dateRangeEnd**: The end date of the period for which statistics are reported. • **roomType**: The type of room for which revenue inventory statistics are reported. • **marketCode**: The market segment associated with the revenue inventory data. • **guaranteeCode**: The reservation type or guarantee code associated with the revenue inventory data. • **revenueCounts**: Detailed counts and statistics related to revenue: – **totalRevenue**: The total revenue generated. – **roomRevenue**: Revenue generated specifically from room bookings. – **foodBeverageRevenue**: Revenue generated from food and beverage sales. – **otherRevenue**: Revenue from other sources. • **occupancyRates**: Statistics related to occupancy: – **occupiedRooms**: The number of rooms occupied. – **availableRooms**: The number of rooms available. – **occupancyPercentage**: The percentage of rooms occupied.

Business Logic	Comments

Sample API request/ response	1. startRevenueInventoryStatisticsProcess
	
	Endpoint:
	
	POST {{HostName}}/inv/async/v1/externalSystems/ {{ExternalSystemId}}/hotels/{{HotelId}}/ revenueInventoryStatistics
	
	Request body:
	{
	"dateRangeStart": "2023-06-23",
	"dateRangeEnd": "2023-06-25",
	"roomTypes": [
	""
	]
	}
	
	Sample Successful Response:
	
	202 Accepted
	
	2. getRevenueInventoryStatisticsProcessStatus
	
	Endpoint:
	
	HEAD {{HostName}}/inv/async/v1/externalSystems/ {{ExternalSystemId}}/hotels/{{HotelId}}/ revenueInventoryStatistics/{{RequestId1}}
	
	Sample Request cURL:
	
	curl --location --head '{{HostName}}/inv/async/v1/ externalSystems/{{ExternalSystemId}}/hotels/ {{HotelId}}/revenueInventoryStatistics/ {{RequestId1}}' \
	--header 'Content-Type: application/json' \
	--header 'x-hotelId: {{HotelId}}' \
	--header 'x-app-key: {{AppKey}}' \
	--header 'Authorization: Bearer {{Token}}' \
	--data ''
	
	Sample Successful Response:
	
	201 Created
	(You are ready to proceed to Step 3)

Business Logic	Comments

	3. getRevenueInventoryStatistics
	
	Endpoint:
	
	GET {{HostName}}/rtp/async/v1/externalSystems/{{ExtSystemCode}}/hotels/{{HotelId}}/rates/hurdles/{{RequestId2}}
	
	Sample Successful (200 OK) Response:
	{
	"revInvStats": [
	{
	"property": "BAA_AUTO",
	"occupancyDate": "2023-06-23",
	"physicalRooms": "290",
	"noShowRooms": "4"
	},
	{
	"property": "BAA_AUTO",
	"occupancyDate": "2023-06-24",
	"physicalRooms": "290",
	"noShowRooms": "1"
	},
	{
	"property": "BAA_AUTO",
	"occupancyDate": "2023-06-25",
	"physicalRooms": "290"
	}
	]
	}

3. Creating Sell Limits using postSellLimitsProcess

The asynchronous operation postSellLimits allows setting maximum room sell limits for a specified date range, room type, or room class in OPERA Cloud. This operation helps RMS manage room inventory effectively, prevent overbooking, and optimize revenue by ensuring accurate and controlled room availability across different sales channels.

Business Logic	Comments
Posting the Sell Limits	The sell limits can be set using specific parameters to control the maximum number of rooms that can be sold for a specified date range. **hotelId**: Specifies the hotel for which sell limits are being set. You can send the specific sell limit details to be posted with the following information: • **sellLimitDate**: The date for which the sell limit is to be applied. • **roomType**: The type of room to which the sell limit applies. • **roomClass**: The class of the room to which the sell limit applies. • **sellLimitCategory**: The category for sell limits, such as house, room type, or room class. • **sellLimitAmount**: The maximum number of rooms that can be sold. • **sellLimitType**: Indicates if the sell limit is a flat number or a percentage. • **bookingChannel**: Specifies the booking channel for which the sell limit applies. • **override**: A flag indicating if the sell limit should override existing limits.
Key Considerations	• The hotelId parameter is essential to identify which hotel's sell limits are being set or updated. • Specify the exact sellLimitDate to ensure accurate implementation and avoid overlaps or gaps in sell limits. • Use the sellLimitCategory to apply limits at the house, room type, or room class level for precise control over inventory. • Set the sellLimitAmount accurately to control the maximum number of rooms that can be sold and optimize room availability. • Define whether the sellLimitType is a flat number or percentage to ensure the limit is applied correctly. • Include relevant roomType and roomClass values to apply sell limits to specific categories of rooms. • Specify the bookingChannel to control which channels the sell limits apply to, optimizing revenue management across different sales channels. • Use the override flag judiciously to determine whether the sell limit should override existing limits, preventing unintended conflicts.
Validation	Validate the applied sell limits through the RMS to ensure they match the intended configurations and do not conflict with other limits or booking policies.

Business Logic	Comments

Sample API request/ response	1. postSellLimitsProcess
	
	Endpoint:
	
	POST {{HostName}}/inv/async/v1/externalSystems/ {{ExtSystemCode}}/hotels/{{HotelId}}/sellLimits
	
	Request body:
	[
	{
	"houseSellLimits": [
	{
	"date": "2024-05-14",
	"amount": 0,
	"flatOrPercentage": "Flat"
	}
	],
	"roomTypeSellLimits": [
	{
	"date": "2024-05-14",
	"amount": 0,
	"flatOrPercentage": "Flat",
	"roomType": "string"
	}
	],
	"roomClassSellLimits": [
	{
	"date": "2024-05-14",
	"amount": 0,
	"flatOrPercentage": "Flat",
	"roomClass": "string"
	}
	]
	}
	]
	
	Sample Successful Response:
	
	202 Accepted
	
	2. getSellLimitsProcessStatus
	
	Endpoint:
	
	HEAD {{HostName}}/inv/async/v1/externalSystems/ {{ExtSystemCode}}/hotels/{{HotelId}}/sellLimits/ {{RequestId1}}

Business Logic	Comments
	Sample Successful (200 OK) Response: 201 Created (You are ready to proceed to Step 3) 3. getSellLimits Endpoint: GET {{HostName}}/inv/async/v1/externalSystems/ {{ExtSystemCode}}/hotels/{{HotelId}}/sellLimits/ {{RequestId2}} Sample Successful Response: 200 OK <The Sell Limits are posted to OPERA Cloud >

Restrictions Management

Restrictions help you manage occupancy and revenue (RevPAR) by defining the conditions under which specific rates, room types, and/or room classes are available to sell for new reservations. For example, you could place a two-night minimum stay restriction on all rooms and rates during a weekend when demand is high, or you could set extended stay durations on certain room types and rates to ensure maximum revenue is achieved over periods of high demand. Restrictions can be set at the house level, meaning that any restrictions placed are applicable to all rooms/room types or based on the channel, rate code, rate category, room type, or room class.

Figure 4-7 Restrictions Management Workflow

Use Cases

Use Case	Description	Operation
Posting a single restriction	You can post a single restriction. You can create a Restriction at various levels, such as house level, rate class, rate category, rate plan, room class, room type, and booking channel code. You can combine restrictions to apply to more than one element. For example, you are able to restrict reservations for a specific room type and rate code combination.	postRestriction

Use Case	Description	Operation
Posting multiple restrictions	You can create up to 1000 restrictions at once with postRestrictions. These can be created at various levels such as house level, rate class, rate category, rate plan, room class, room type, and booking channel code. You can combine restrictions to apply to more than one element. For example, you are able to restrict reservations for a specific room type and rate code combination.	postRestrictionsProcess
Retrieving restrictions for a hotel for a given date range	You can retrieve the Restrictions for a hotel in a given date range. The search criteria can include filter codes or restriction control types, date range, and the hotel information.	getRestrictionsByDateRange

1. Posting a single restriction using postRestriction

The synchronous operation postRestriction is used to set or update one restriction at a time for a hotel. Restrictions can be set at the house level, meaning that any restrictions placed are applicable to all rooms/room types or based on the channel, rate code, rate category, room type, or room class.

Business Logic	Comments
Posting a single restriction	The restrictions can be set using specific parameters to control room availability and booking patterns. **hotelId**: Specifies the hotel for which restrictions are being set. You can send the specific restriction details to be posted with the following information: • **restrictionStatus**: Defines the type and specifics of the restriction. – **code**: Type of restriction (for example, MinimumLengthOfStay). – **lOS1 to lOS7**: Flags indicating the length of stay restrictions for 1 to 7 nights. • **ratePlanCodes**: An array of rate plan codes to which the restriction applies. • **roomClasses**: Classes of rooms affected by the restriction. • **bookingChannels**: Channels through which the booking can be made. • **roomTypes**: Types of rooms affected by the restriction. • **ratePlanCategories**: Categories of rate plans affected by the restriction. • **Sunday to Saturday**: Flags indicating the days of the week the restriction applies. • **start**: The beginning date for the restriction. • **end**: The ending date for the restriction.

Business Logic	Comments
Key Considerations	• The hotelId parameter is essential to identify which hotel's restrictions are being set or updated. • Specify the exact start and end dates for which the restrictions are to be applied to ensure accurate implementation and avoid overlaps or gaps. • Define restrictionStatus clearly to indicate the type of restriction being applied (for example, MinimumLengthOfStay), ensuring it aligns with the hotel's booking policies. • Set lOS1 to lOS7 flags accurately to control the allowed booking durations and optimize room utilization. • Include ratePlanCodes, roomClasses, and roomTypes to apply restrictions to specific rate plans, room classes, and room types, ensuring precise control over availability and pricing. • Specify the bookingChannels and ratePlanCategories to control which booking channels and rate plan categories the restrictions apply to, optimizing revenue management. • Set day-of-week flags (Sunday to Saturday) appropriately to ensure the restrictions are applied on the correct days.
Validation	Validate the applied restrictions through the RMS to ensure they match the intended configurations and do not conflict with other restrictions or booking policies.

Business Logic	Comments

Sample API request/ response	postRestriction
	
	Endpoint:
	
	POST {{HostName}}/par/v1/hotels/{{HotelId}}/ restrictions
	
	Request body:
	{
	"restriction": {
	"restrictionStatus": {
	"code": "Closed",
	"lOS1": false,
	"lOS2": false,
	"lOS3": false,
	"lOS4": false,
	"lOS5": false,
	"lOS6": false,
	"lOS7": false
	},
	"ratePlanCodes": [
	"RACK"
	],
	"roomClasses": [],
	"bookingChannels": [],
	"roomTypes": [],
	"ratePlanCategories": [],
	"hotelId": "SH765",
	"sunday": true,
	"monday": true,
	"tuesday": true,
	"wednesday": true,
	"thursday": true,
	"friday": true,
	"saturday": true,
	"start": "2023-11-07",
	"end": "2023-11-07"
	}
	}
	
	Sample Successful Response:
	
	201 Created
	The Restrictions are posted to OPERA Cloud

2. Posting multiple restrictions using postRestrictionsProcess

The asynchronous operation postRestrictionsProcess is used to set or update multiple restrictions for a hotel for a given date range. The postRestrictionProcess endpoint is designed for setting or updating restrictions on reservations in a hotel system. These restrictions can

include rules such as minimum length of stay, specific rate plans, room types, and other conditions that control how rooms can be booked.

Business Logic	Comments
Posting multiple restrictions	The restrictions can be set using the following parameters to control room availability and booking patterns. • **hotelId**: Specifies the hotel for which restrictions are being set. You can send the specific restriction details to be posted with the following information: • **restrictionStatus**: Defines the type and specifics of the restriction. – **code**: Type of restriction (for example, MinimumLengthOfStay). – **unit**: Unit of measurement for the restriction. – **lengthOfStay1 to lengthOfStay7**: Flags indicating the length of stay restrictions for 1 to 7 nights. • **ratePlanCodes**: An array of rate plan codes to which the restriction applies. • **roomTypes**: An array of room types affected by the restriction. • **ratePlanCategories**: An array of rate plan categories affected by the restriction. • **seasonCode**: Code identifying the season during which the restriction applies. • **blockId**: Identifier for the block of rooms affected by the restriction. – **id**: Specific block ID. • **bookingChannelOnRequest**: Specifies the booking channel on request for the restriction. • **dateRangeStart**: The start date for the restriction. • **dateRangeEnd**: The end date for the restriction. • **Day of Week Flags**: Flags indicating if the restriction applies on a particular day of the week. • **yieldStatus**: Indicates whether the restriction is yieldable or not.
Key Considerations	• Specify the exact dateRangeStart and dateRangeEnd to ensure accurate implementation and avoid overlaps or gaps. • Define restrictionStatus clearly to indicate the type of restriction being applied, such as MinimumLengthOfStay. • Set lengthOfStay flags (lengthOfStay1 to lengthOfStay7) accurately to control the allowed booking durations and optimize room utilization. • Include ratePlanCodes and roomTypes to apply restrictions to specific rate plans and room types, ensuring precise control over availability and pricing. • Specify the seasonCode and blockId to manage seasonal and block-specific restrictions. • Use bookingChannelOnRequest to define the applicable booking channels for the restrictions. • Set day-of-week flags (Sunday to Saturday) appropriately to ensure the restrictions are applied on the correct days. • Specify the yieldStatus to indicate whether the restriction is yieldable or not.
Validation	Validate the applied restrictions through the RMS to ensure they match the intended configurations and do not conflict with other restrictions or booking policies.

Business Logic	Comments

Sample API request/ response	1. postRestrictionsProcess
	
	Endpoint:
	
	POST {{HostName}}/par/async/v1/externalSystems/ {{ExtSystemCode}}/hotels/{{HotelId}}/restrictions
	
	Request Body:
	{
	"restriction": {
	"restrictionStatus": {
	"code": "Closed",
	"lOS1": false,
	"lOS2": false,
	"lOS3": false,
	"lOS4": false,
	"lOS5": false,
	"lOS6": false,
	"lOS7": false
	},
	"ratePlanCodes": [
	"RACK"
	],
	"roomClasses": [],
	"bookingChannels": [],
	"roomTypes": [],
	"ratePlanCategories": [],
	"hotelId": "SH765",
	"sunday": true,
	"monday": true,
	"tuesday": true,
	"wednesday": true,
	"thursday": true,
	"friday": true,
	"saturday": true,
	"start": "2023-11-07",
	"end": "2023-11-07"
	}
	}
	
	Sample Successful Response:
	
	202 Accepted
	
	2. getRestrictionsProcessStatus
	
	Endpoint:
	
	HEAD {{HostName}}/par/async/v1/externalSystems/

Business Logic	Comments
	{{ExtSystemCode}}/hotels/{{HotelId}}/restrictions/ {{RequestId1}} Sample Successful Response: 201 Created (You are ready to proceed to Step 3) 3. getRestrictions Endpoint: GET {{HostName}}/par/async/v1/externalSystems/ {{ExtSystemCode}}/hotels/{{HotelId}}/restrictions/ {{RequestId2}} Sample Successful Response: 200 OK <The Restrictions are posted to OPERA Cloud>

3. Retrieving restrictions for a hotel for a given date range

The synchronous operation getRestrictionsByDateRange is used to retrieve detailed information about the restrictions applied to a hotel's booking process within a specified date range. This information is crucial for understanding and managing booking limitations, ensuring compliance with hotel policies, and optimizing room inventory. You can utilize the data for operational planning, revenue management, and enhancing the guest booking experience.

Business Logic	Comments
Retrieving restrictions	The restrictions data can be requested using restrictionSearchCriteriaStartDate and end to fetch the restrictions for a specified date range. • **restrictionSearchCriteriaStartDate**: The beginning date to filter restrictions. • **end**: The ending date to filter restrictions. Other critical parameters include: • **hotelId**: Specifies the hotel for which restrictions are being queried. • **restrictionStatus**: Filters restrictions based on their current status. • **ratePlanCodes**: Filters restrictions based on the rate plan codes applied. • **roomTypes**: Filters restrictions based on the type of room. • **seasonCode**: Filters restrictions based on the season code. • **blockId**: Retrieves a specific block by its unique ID. • **bookingChannelOnRequest**: Specifies the booking channel for which the restrictions apply.

Business Logic	Comments
Key Considerations	• Specify the exact restrictionSearchCriteriaStartDate and end dates to ensure accurate and comprehensive retrieval of restrictions. • Use restrictionStatus to filter results based on the current status of the restrictions, ensuring you get relevant data. • Include ratePlanCodes to filter restrictions based on specific rate plans, which helps in obtaining detailed and specific information. • Specify roomTypes to filter restrictions based on the types of rooms, ensuring precise control over the data retrieved. • Use seasonCode to manage seasonal restrictions and ensure that the data reflects the correct seasonal policies. • Include blockId to retrieve specific blocks by their unique ID, which is helpful for managing group reservations or specific events. • Use bookingChannelOnRequest to filter restrictions based on the booking channel, ensuring that the retrieved data is relevant to specific sales channels.
Key Response Data Elements	The key response data elements include: **RestrictionsByDateRange**: • **hotelId**: The ID of the hotel for which restrictions are being retrieved. • **hasMore**: Indicates if there are more restrictions beyond the current response. **RestrictionSets**: • **restrictionControl**: Details of the restriction control. – **house**: Indicates if the restriction applies to the entire house. – **roomClass**: The class of the room affected by the restriction. – **ratePlanCategory**: (Optional) The rate plan category affected by the restriction. • **restrictionStatus**: Status details of the restriction. – **code**: The type of restriction (for example, MaximumLengthOfStay, ClosedForDeparture). – **unit**: (Optional) The unit of the restriction, such as the number of nights. • **actualTimeSpan**: The time span during which the restriction is applied. – **startDate**: The starting date of the restriction. – **endDate**: The ending date of the restriction. • **onRequest**: Indicates if the restriction is on request. • **start**: The start date of the restriction. • **end**: The end date of the restriction. • **Day of Week Flags**: Flags indicating if the restriction applies on a particular day of the week.

Business Logic	Comments

Sample API request/ response	getRestrictionsByDateRange
	
	GET {{HostName}}/par/v1/hotels/{{HotelId}}/ restrictions? hotelId={{HotelId}}&restrictionSearchCriteriaStartDat e={{StartDate}}&end={{EndDate}}
	
	Sample Successful (200 OK) Response:
	
	{
	"restrictionsByDateRange": {
	"restrictionsByDateRange": {
	"restrictionSets": [
	{
	"restrictionControl": {
	"house": true
	},
	"restrictionStatus": {
	"code": "Closed"
	},
	"actualTimeSpan": {
	"startDate": "2023-11-06",
	"endDate": "2023-11-06"
	},
	"onRequest": false,
	"start": "2023-11-06",
	"end": "2023-11-06",
	"sunday": true,
	"monday": true,
	"tuesday": true,
	"wednesday": true,
	"thursday": true,
	"friday": true,
	"saturday": true
	},
	{
	"restrictionControl": {
	"house": false,
	"ratePlanCode": "AMMAR1"
	},
	"restrictionStatus": {
	"code": "Closed"
	},
	"actualTimeSpan": {
	"startDate": "2023-11-06",
	"endDate": "2023-11-07"
	},
	"onRequest": false,
	"start": "2023-11-06",
	"end": "2023-11-06",
	"sunday": true,
	"monday": true,

Business Logic	Comments
	"tuesday": true, "wednesday": true, "thursday": true, "friday": true, "saturday": true }], "hotelId": "SH765", "hasMore": false } } }

5

References

Reference	Description	Link
Oracle Hospitality APIs	You can access examples of the currently supported Asynchronous API operations categorized by Module in the provided Postman collection.	Oracle Hospitality Postman Workflow collections
Configuring External Systems	All asynchronous operations require you to create an external system. Oracle Hospitality OPERA Cloud requires this information to identify the recipient of the request. This document explains the step-by-step procedure to create an external system.	Configuring External Systems
OPERA Cloud Rate Code User Guide	Rate Codes form the core of OPERA Cloud reservation and rate process. Rate codes are used to define the various prices for each room type over a particular date range or rate season. Refer to this document for additional details.	About Rate Codes
Rate Management User Guide	Rate Management provides all the tools you need to effectively define and manage the rate structures for your hotel. Some of the things you can set up in Rate Management include rate codes, rate classes, rate categories, display sets, rate strategies, and yield management integration. Refer to this document for additional details.	About Rate Management
OPERA Cloud Sell Limits User Guide	Sell Limits let you control room availability at your property. You can configure sell limits from the Property Availability screen in the Inventory section. Refer to this document for additional details.	About Sell Limits
OPERA Cloud Hurdle Rates User Guide	Revenue Management (yield) solutions calculate the actual hurdle rates that OPERA Cloud compares against. Refer to this document for additional details.	About Hurdle Rates
OPERA Cloud Restrictions User Guide	Restrictions help you manage occupancy and revenue (RevPAR) by defining the conditions under which specific rates, room types, and/or room classes are available to sell for new reservations. Refer to this document for additional details.	About Restrictions
OHIP Digital Learning	The learning path will provide an overview of the benefits, highlight features, and provide instruction on the fundamentals of OHIP.	Oracle Hospitality Integration Platform (OHIP) for Integrators Learning Path

6

Key Terminologies

Term	Description
External System	All Revenue Management Systems (RMS) APIs require you to create an external system. This is so Oracle Hospitality OPERA Cloud knows for whom to process the request.
Throttling Limits	The Oracle Hospitality Revenue Management Systems (RMS) APIs are throttled. Throttling limits the number of concurrent calls to OPERA Cloud to avoid impacting day-to-day hotel operations. If more than 50 requests per second are received for a single OPERA Cloud environment, throttling helps maintain the day-to-day running of the hotel while still enabling API traffic.
Business Events	A business event is an event (that is, an update) that happens to a resource, for example, a reservation. Rather than getting the reservation repeatedly, subscribing to the Update Reservation event informs you when this or other reservations are changed. There are two methods for consuming events from Oracle Hospitality APIs: Polling and Streaming.
Streaming API	When you consume business event data from OPERA Cloud, you can choose which events an application will receive. These are the events to which the application subscribes. You can subscribe to business events from a specific environment, but approval from the environment owner is required. Additional information can be found on the Oracle Hospitality Innovation Week page.
Polling API	You can consume business event data from OPERA Cloud by polling, which refers to retrieving updates or new data from the server at regular intervals. Additional information can be found on the Oracle Hospitality Innovation Week page.
Application Key	A unique application key is created for each application upon registering an application in the Oracle Hospitality Developer Portal.
Scope	A static variable that represents the authorization scope and permissions granted to the client for accessing the APIs from OHIP.
Token	A header “Authorization” that includes the oAuth token obtained through Authenticating to Oracle Hospitality Revenue Management Systems (RMS) APIs.